ROUTING AND SCHEDULING WITH TIME WINDOWS: MODELS AND ALGORITHMS FOR TRAMP SEA CARGOS AND RAIL CAR-BLOCKS

A Thesis
Presented to
The Academic Faculty

by

Aang Daniel

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Industrial and Systems Engineering

Georgia Institute of Technology
December 2006

ROUTING AND SCHEDULING WITH TIME WINDOWS: MODELS AND ALGORITHMS FOR TRAMP SEA CARGOS AND RAIL CAR-BLOCKS

Approved by:

Professor Faiz Al-Khayyal, Advisor
School of Industrial and Systems Engineering
Georgia Institute of Technology

Professor Earl Barnes
School of Industrial and Systems Engineering
Georgia Institute of Technology

Professor Ellis Johnson
School of Industrial and Systems Engineering
Georgia Institute of Technology

Professor I. A. Karimi
Department of Chemical and Biomolecular Engineering
University of Singapore

Professor Joel Sokol
School of Industrial and Systems Engineering
Georgia Institute of Technology

Date Approved: 13 November 2006

In loving memory of my beloved mother...

ACKNOWLEDGEMENTS

"The fear of the Lord is the beginning of knowledge" (Proverbs 1:7). Let me begin by thanking God, for he is the beginning of everything. He was also the one who gave me strength and courage to finish my Ph.D. at Georgia Tech. In the same time, I would like to thank the blessed virgin Mary for her prayers throughout my study. My academic journey is truly a journey of faith.

I am thankful to my advisor, Dr. Faiz Al-Khayyal, for his kindness, encouragement, patience, and advice ever since the first day of my life at Georgia Tech. I would like to specially acknowledge Dr. Ellis Johnson for his counsel and guidance that have been invaluable to me. I also would like to thank Dr. Joel Sokol, Dr. Earl Barnes, and Dr. Karimi for serving on my dissertation committee and always being there for me.

I would like to express my gratitude to the Indonesian Catholic Community (Komunitas Katolik Indonesia) in Atlanta for all the prayers and the great experience of being part of the community. To my fellow students, especially Seunghyun, Ethan, Yaxian, and all members and alumni of the Indonesian Student Association (ISA) at Georgia Tech for the joy and friendship throughout my study. Also, to my best friends, Audie and Shanti, for convincing me that I was smarter than I actually am.

Finally, I would like to thank my family. I would like to thank my Dad, a remarkable man with a big heart, and my Mom in heaven, a woman whose most gentle soul embraces me each day of my life. I am deeply grateful to my wonderful wife, Elisa. Without her unwavering and unconditional support I would never have completed my study. Also, to my little angle, Avila, for being such a blessing from God.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

This research introduces a new formulation to solve routing and scheduling problems, with the main application in answering an actual problem faced by a sea-cargo shipping company in South-East Asia. Our formulation better lends itself to extensions and related applications. The model can also be adapted to the solution of train routing and scheduling problems faced by a railroad company in the United States.

For the work in sea-cargo routing and scheduling, we focus our methodology on the *tramp* shipping operation. Tramp shipping is a demand-driven type of shipping operation. Tramp ships typically do not have fixed schedules, but are routed based on the pickup and download locations of profitable service requests. The problem is that given a set of products distributed among a set of ports, with each product having a pickup time window, download time window, and destination port, find the schedule for a fleet of ships that maximizes profit for a specified time horizon. The problem is modelled as a Mixed Integer Non-Linear Programming (MINLP) problem. We use quite a number of bilinear constraints that are linearized by taking advantage of the linearization scheme based on a convex and concave envelope approach. This allows us to reformulate our model into an *equivalent* Mixed Integer Linear Program (MILP).

We introduce three heuristic methods for solving sea-cargo routing and scheduling problems. The first heuristic employs the idea of a rolling horizon concept. Here we divide the time horizon into smaller time periods. We solve the earlier period and carry over the decision to the next period. The second heuristic utilizes the optimal solutions to one-ship problems. Here we permanently assign cargos that appear in

only one ship and determine which ship should carry cargos that appear in more than one ship by creating a priority list. Based on the list, the model is again used to break any ties among competing ships. The third heuristic utilizes the set packing approach. Columns are created *a priori* based on cargo combinations that can be served by each ship. Our model is used to check the feasibility of a cargo combination and give us the corresponding route and optimal profit. Cargos are added to the system using a predefined priority list. If we allow the method to find all possible cargo combinations that can be served by each ship, we can eventually find the optimal solution. Extensive computational results are presented to show how good these heuristics perform in solving randomly generated problems. Generally, the proposed heuristic methods find solutions that are within at least 80% of the optimal profit reasonably fast.

We also exploit the special structure enjoyed by our formulation and introduce an upper-bounding problem to our model. Our computational results show that the LP relaxation of the upper bounding problem gives a bound that is $2 - 10\%$ times stronger than that given by the LP relaxation of the original problem. Also, those bounds are computed within 3 to 5 times faster than the time spent to find the LP relaxation of the original problem.

With a little modification, the model for our sea-cargo routing and scheduling problem is readily extendable to reflect other practical needs in sea cargo operations. We introduce the notions of soft time windows with penalty and inter-ship cargo-transfer as extensions to our problem to show the advantages of our formulation of the problem. Both of these extensions involve extensive time aspects. In the soft time-window extension, we allow pickups and downloads to happen outside the predefined time-windows by imposing some penalty terms. This is where the time aspects come into play. In the cargo-transfer extension, we allow a cargo to be picked-up and downloaded by different ships. We present some computational results on

these extensions. The results show that more savings can be generated by including soft time-windows and inter-ship cargo-transfer capabilities to the model. Soft time-windows and cargo-transfer also allow more realistic situations in sea-cargo operation.

The other part of our work is the application of the same modeling technique to solve a train routing and scheduling problem faced by a real railroad company in the United States. Train routing and scheduling problems involve a lot of time aspects since one of the goals is to have optimal train schedules and timings.

The train scheduling problem is a very large-scale network optimization problem which typically calls for trillions of decision variables. One very important concept in train scheduling problems is the *block* concept. Block concept arises in the context that a railroad serves thousands or millions of shipments from their origins to respective destinations. A typical shipment consists of a set of *cars* having a common origin and destination. To reduce the handling of individual shipments as they travel, a set of shipments is grouped together as a *block*. Previous research in train routing and scheduling problems typically calls for separating the problem into several phases. Train route design problems and block-to-train assignment problems are usually solved separately using an iterative procedure with feedbacks from one phase to another. Our model solves the routing and timing/scheduling problems altogether and proposes a new approach in solving the train routing and scheduling problems.

Given a set of blocks to be carried from their origins to their destinations, our goal is to decide how many trains are needed, construct train routes and schedules, and determine block-to-train assignments. We want the solution that minimizes the number of block transfers (block swaps) between trains, the number of trains used to serve all blocks, and some other efficiency measures used in train operations. Some computational experiments from real railroad data are presented and the model successfully provides a different approach in solving train routing and scheduling problems by giving not only the train routes but also the optimal timings/schedules simultaneously.

We anticipate that our models will provide alternate modeling techniques to solve routing and scheduling problems found in other transportation industries.

CHAPTER I

INTRODUCTION

Ships and trains are two very important modes of transportation. They are both considered to be the most cost effective transportation modes. Both operate long hours and carry larger and heavier products compared to other transportation modes. Ship and train industries are also rich with opportunities to implement optimization models. A small improvement may result in millions of dollars in savings. Both industries were very conservative in accepting and applying optimization models in their business practices. From a research point of view, ship and train industries have historically received little research attention. However, as researchers realized the importance of cost savings and performance improvements in these industries, a growing body of advances concerning optimizations in these two industries has increasingly appeared in the operations research literature.

This dissertation focuses on the introduction of a modeling technique that can better represent practical situations in answering routing and scheduling problems in the ship and train industries. The model is a Mixed Integer Nonlinear Program (MINLP) that can be reformulated as an equivalent Mixed Integer Linear Program (MILP).

1.1 Ship routing and scheduling

One major transportation mode of international trade is sea cargo shipping. Sea cargo transportation is the logistics workhorse in global supply chains. Approximately 70% of the value of all goods are transported worldwide by sea [49], but relatively little work has been done on ship routing and scheduling. Recent United Nation's Shipping and World Trade Reports also show that approximately 90% of the world

trade tonnage is transported worldwide by sea. Worldwide traffic is expected to grow at an average rate of between 3% to 6% each year until at least 2024. This rate applies for all types of cargo categories (dry and liquid bulk, container, and general cargo). The volatility of global fuel costs has significantly reinforced the need for optimal ship schedules that minimize ship operational costs.

Although the trend shows a significant increase in the usage of ships, the ocean shipping companies are faced with ever-increasing competition among players, where profit margins are squeezed to a very minimum level. Ships are capital-intensive and a ship may cost tens of thousands of dollars a day to operate. Whether it is a company that owns and manages a fleet of ships, or a ship leasing company that manages a fleet and is hired by either a shipper or a third party logistics provider, the cost incurred by a ship's route and schedule directly affects the cost effectiveness of global supply chain systems.

Optimal assignments of cargos and optimal schedule generations are complex combinatorial problems. A better schedule that saves 5% of the current operating costs can result in millions of dollars in savings annually.

Ship routing and scheduling problems are somewhat different from those of other transportation modes because of the time spent by a ship to travel from one destination to another. Ships generally spend a great amount of time traveling from one destination to another. Even though a ship may spent time waiting in a port, the total time spent in a port is far less than the traveling time. Typical travel time ranges from as short as several hours to as long as several months.

A ship's route and schedule are also very sensitive to weather and ocean environment changes that are indeed unpredictable. Consequently, a ship may frequently change its schedule to avoid dangerous weather conditions. Hence, the set of routes has to be reoptimized many times.

Shipping companies usually maintain a large variety of fleets that operate 24 hours a day. The schedule for each ship voyage may be as long as 4 months and the

destination might change while a ship is in the middle of its journey. A ship can only visit some of the ports since its size might be incompatible with some port terminals. To some extent, aircraft operations are similar to ship operations. They both pay port fees and operate internationally. Their daily operating cost is very large. But mostly aircrafts transfer passengers or packaged goods, while ships load mostly liquid and dry bulk cargos, or many types of containers.

Compared to other transportation modes (such as train, truck, aircraft), there has been far less attention given to ship. One of the possible reasons is that even though a ship is a major transportation mode worldwide, it is commonly seen by people in their daily lives. Another reason is that ship-related industries are conservative and not very open to new ideas. Many companies are small and owned by a family. People employed in ship routing and scheduling departments are most likely those with years of experience so that they are reluctant to accept new technologies for fear that they might not be needed anymore. Also, since most ship companies are small and family owned, they are less likely to devote some of their profits to fund research. However, as the trend of mergers, acquisitions, and joint-ventures increases in the shipping industry, the role of optimization in this area will become more significant.

There are three types of ship operations. The first one is a classical industrial shipping operation, which tries to minimize the sum of all costs generated by all ships while ensuring all cargos are transported from origin to destination. Usually this problem is faced by a multi-national company that has to transport goods within the company to fulfill demands in many ports. But today more and more companies focus on their business and outsource this kind of transportation problem to a third party.

The second type is a tramp shipping operation, which is the focus of the work in this thesis. Tramp shipping operates like a taxi service; whenever there is profitable cargo to load, a ship is dispatched to pickup the cargo using a route and schedule that tries to maximize its profit. Some of the tramp shipping companies also engage

in shipping contracts. Little work has been done in this area because of the large number of small operators in the tramp market. But in this era of mergers and acquisitions, tramp shipping is now an interesting research topic. Also, as transportation outsourcing becomes a trend, the demand for tramp shipping is bound to increase. The third shipping operation is liner shipping, which is quite different from the two described above because it involves decisions at different planning levels. This is beyond the scope of this study.

1.2 Previous work on ship routing and scheduling

A classic attempt to determine an optimal schedule for a fleet of ships in tramp operation was pioneered by Appelgren [6]. He used a relaxed set partitioning approach and applied the Dantzig-Wolfe decomposition method to solve the problem. The solution strategy could not guarantee an integer solution, but it managed to solve problems with practical sizes.

Quite a number of papers discussed the scheduling of a fleet of ships in industrial operations. Ronen [54] examined the short-term scheduling of a fleet of bulk ships: each starts in a single loading port and ends at the same port after visiting a sequence of download ports. The objective is to minimize the cost and the solution strategy proposed is the set partitioning method. The method managed to solve small problems to optimality. Scott [59] also studied industrial shipping problems where oil and other petrochemical products are shipped from a refinery station to several processing centers using heterogeneous multi-compartment ships. The solution strategy proposed is Lagrangian relaxation and a refined version of Bender's decomposition method to avoid solving a large integer model. Recently, Hwang [30] formulated a model for finding a minimum cost routing in a network for a heterogeneous fleet of ships engaged in pickup and delivery of several liquid bulk cargos in an industrial operation environment. The problem is frequently encountered by maritime chemical transport companies, including oil companies serving an archipelago of islands. The

model decides how much of each product should be carried by each ship from supply ports to demand ports, such that the inventory level of each product in each port is maintained.

There are also studies in industrial shipping operations inspired by problems faced by the US Navy. Psaraftis [50] studied how to allocate cargos to ships such that all cargos arrive at respective destinations as planned. The algorithm proposed uses a rolling horizon strategy where the cargos at the front-end time horizon are permanently assigned to be carried but cargos available at a later time are assigned tentatively. Fisher and Rosenwein [24] studied the scheduling of ships serving less-than-shipload bulk cargos with time windows for the US Navy's Military Sealift Command. The goal is to minimize the cost incurred, and the proposed solution strategy is a set partitioning approach with total schedule enumeration.

Even though during the last decades there has been a shift from industrial shipping to the tramp sector, most contributions in general are still for industrial shipping. Only a few studied the tramp sector. A typical tramp ship scheduling problem is presented in Appelgren [6]. Kim and Lee [37] developed a prototype of decision-support system for scheduling a fleet of ships serving a bulk trade. The underlying problem is formulated as a set packing problem. They also present an algorithm to generate all feasible columns (feasible routes) a priori. For works where shipping companies operate their ships in both industrial and tramp mode, see Bausch et al. [9], Christiansen [16], and Sherali et al. [63].

There are many solution approaches that have been proposed to solve ship routing and scheduling problems. However, in the survey presented by Christiansen, Fagerholt, and Ronen [14] it is observed that 40% of the reviewed papers use a column generation approach. The two main common solution approaches used are the Dantzig-Wolfe decomposition approach (for example [6], [16]) and the set partitioning approach with some or all columns generated *a priori* (for example [9], [22], [24]).

Christiansen et al. [13] provide an extensive review on maritime transportation

and all developments in the last decade. For complete reviews of the work in ship routing and scheduling, readers are also referred to survey papers by Ronen ([55], [53], and [12]).

1.3 Pickup and Delivery Problem

Our ship problem can be treated as a multi-vehicle pickup and delivery problem (PDP) which has been studied by numerous authors. Savelsbergh and Sol [56] provide a thorough review of the work in this area. The most commonly studied variant of the PDP is the Pickup and Delivery Problem with Time Windows (PDPTW).

PDPTW is a problem that is concerned with the optimal route construction to satisfy transportation requests, each requiring a pickup and a download at specific locations, under vehicle capacity and time window requirements. There is also a pairing constraint embedded since each request has to be picked up and downloaded by the same vehicle. PDPTW usually requires the vehicles to return to their original location while in our problem, the final paths for the vehicles are open paths, ending at the last scheduled delivery. It is not mandatory to have the ships return to their base location.

Several heuristic and exact methods exist for the PDP. Optimization based solution usually formulates problems as integer programs. There are two types of models in the literature. The first one tries to find the best feasible path by evaluating the arcs to be included in the schedule. The other one follows a two-phase set partitioning method, where phase I generates a set of candidate schedules and phase II solves a set partitioning model to construct the final schedule. Problem size is often reduced by decomposing the problem or generating possible schedules heuristically.

1.3.1 Optimization based approach for Pickup and Delivery Problem

An exact solution method for PDP was first proposed by Psaraftis [51]. A dynamic programming approach is developed with computational results shown for up to 9 customers. Little et al. [43] develop a branch and bound approach to answer PDP

problems. The work was extended by Kalantari et al. [34] where the tours are divided into some manageable subsets. Computation results are reported for up to 31 customers. Fischetti and Toth [23] apply a branch-and-bound algorithm that employs an additive bounding approach, which bounds based on assignment relaxation, the shortest spanning 1-arborescence relaxation, as well as disjunctions and variable decomposition.

Lu and Dessouky [44] apply a branch-and-cut algorithm based on a 0-1 integer programming formulation. Dumas, Desrosiers, and Soumis [20] present a set partitioning approach to answer the PDPTW problem and a forward dynamic programming approach to generate columns and solve the problem to optimality. Columns are generated as needed by solving a constrained shortest path problem. Generated columns are then used to solve the linear relaxation of the set partitioning problem, then a branch and bound approach is used to obtain an integral solution. The constrained shortest path problem is solved by forward dynamic programming.

Psaraftis [50] also developed an algorithm for solving multiple-vehicle problems in which the vehicles are eventually ships. The capacity of a port is also considered in order to avoid idle times in the ports waiting for loads being picked-up or downloaded. The algorithm uses the rolling horizon principle. Based on the pickup time windows, the algorithm tries to serve cargos with earlier pickup times and gradually assigns more and more cargos permanently to the available ships. In earlier stages, cargos with later pickup time windows are assigned tentatively. The tentative assignment of loads to ships is calculated in two phases. First, the utility u_{js} of assigning cargo j to ship s is calculated for each cargo j and ship s. This utility measurement is a complicated function measuring the assignment's effect on the delivery time of cargo j and all other cargos assigned to ship s, the efficiency of the use of ship s, and port resources. In the second phase, an assignment problem is solved, where the user has to specifically specify the maximum number of cargos that can be assigned to each ship.

1.3.2 Heuristic based approach for Pickup and Delivery Problem

The PDP is a generalization of the Vehicle Routing Problem (VRP), which is a generalization of the Traveling Salesman Problem (TSP). This makes PDP an NP-hard problem in the strong sense. Hence many authors focus their research on heuristic methods.

Most of the work on PDP takes time windows into account. Van der Bruggen et al. [68] use a two-phase local search algorithm for the PDPTW. Both phases use a variable-depth procedure and an embedded arc-exchange algorithm. Sexton and Bodin [60], [61] apply Benders' decomposition method to a mixed binary nonlinear formulation that solves the scheduling and routing components of the PDPTW separately. Jaw et al. [31] propose a heuristic that generates routes by feasibility and cost-based insertion selection. Savelsbergh and Sol [57] apply a sophisticated column management scheme to a branch-and-price algorithm in the development of a decision support system for a European road transportation firm.

Nanry and Barnes [46] develop a tabu search heuristic, where, to generate solutions, origin and destination locations may either be inserted from one route to another or be swapped with an origin-destination pair of another route. Solution quality is improved or feasibility is gained by inserting individual origins or destinations forward or backward within a route since the capacity constraint might be violated by insertion or swap. Toth and Vigo [67] also discuss the tabu search-related heuristic.

1.4 Train routing and scheduling

Another important major in-land transportation mode is rail. The railroad industry is also very rich in terms of problems that can be modeled using mathematical optimization techniques. However, the related literature in train routing and scheduling has shown a slow growth and most contributions have dealt with simplified models that fail to reflect railroad business practices [17]. But after decades of hunkering

down and cutting costs, railroads over the past few years have begun acknowledging that their future depends to a large extent on their ability to provide service that is competitive with other modes, notably trucking, in terms of reliability and customer satisfaction. This is also the reason many researchers currently are turning their attention to railroad problems.

The development of optimization models for train routing and scheduling was hindered by the large size and the high difficulty of the problems. So, practical implementations of optimization models often had limited success, which deterred both researchers and practitioners from pursuing the effort. However, rail has increasingly gained reputation as the most economical in-land mode of transportation and their stock indices have become stronger in recent years. Strong competition between railroad companies, mergers and acquisitions, privatization, increasing computer speed and data management capability are factors to increased research interest in this area.

Now we will begin explaining some railroad terms that will be used throughout the thesis. Demands for railroad are generally expressed as number of *cars* to be transferred from an origin to a destination. The railroad transportation scheduler has to determine a trip plan for the cars to follow. For every origin-destination pair, the corresponding demand can be shipped directly or indirectly. When the demand is a high priority one, a scheduler might assign a train to carry it from the origin all the way to the destination. When the demand is a low priority one that does not require a direct train, it may experience some delays. Either the traffic is consolidated and routed through intermediate transfer points, or the cars have to wait at the originating station until sufficient cars has been accumulated.

To benefit from economies of scale and to make the scheduler perform his job easier, cars are grouped into *blocks*. A block might contain cars with different origins and destinations but share some portion of their journeys. Thus, there is also an origin and a destination associated to a block. Cars might reach their destination by traveling on a sequence of blocks. There are optimization models developed to

provide an optimal blocking plan and how to assign cars into blocks. However, these are beyond the scope of this thesis. The next thing to do after we have the blocking plans, which is within the scope of our work, is to build a train plan that will carry all blocks from their origins to destinations.

A train travels from a train-origin to a train-destination and carries blocks along the way. When a train passes through an intermediate transfer point, it may drop or pickup blocks of cars. A block dropped by an inbound train might be transferred to another to continue its journey. A block might also switch trains several times before reaching the final destination.

The problem for train routing and scheduling lies in determining the best train plan to carry all blocks from origins to destinations. How many trains are needed, what route each train should follow, what blocks to carry, are some of the routing decisions that a scheduler should decide. A scheduler has to decide the timetables for each trains. Moreover, daily trains are preferred to ease operational issues. It means that a railroad company would rather have the same daily trains operating throughout the week. Train frequency, train arrival and departure times in each stop are some of the scheduling decisions.

1.4.1 Previous work in train routing and scheduling

Train routing and scheduling is a very complex problem. Thus, sequential approaches are often adopted. For example, train routings are developed first and train timetables are decided last. Operating plans are usually for 6 months to 1 year, but weekly and daily adjustments are usually made to account for demand variability.

Most optimization models for train and block routing are defined over a network whose nodes represent origins, destinations, or intermediate transfer points. The arcs represent existing or possible train connections between these points. One of the first efforts to integrate multiple components of the freight routing problem is credited to Assad [7] who proposed a multi-commodity network flow model for train routing and

scheduling that incorporates some level of interaction between routing and activities in train intermediate terminals.

Crainic et al. [18] propose a model and a heuristic for a train routing problem. The model is a nonlinear, mixed integer, multi-commodity flow problem that deals with the interactions between blocking and train routing decisions. They use a decomposition scheme that iterates between blocking and routing until the best improvement in the objective function is attained. The objective is to minimize the sum of operational cost and delay costs associated with block routing and train services.

Haghani [26] proposes a formulation and a solution method for a combined train routing and scheduling with empty car distribution problem. The model deals with temporary demand variability, providing empty car distribution decisions as well as the optimal time interval for consecutive train services between pairs of transfer nodes. To account for demand variations from period to period, each transfer node is replicated a certain number of times in a time-space network, depending on the period length and the planning horizon. The resulting model is a mixed integer problem with nonlinear objective function and linear constraints. It is solved using a heuristic decomposition approach that exploits the structure of the problem by solving an integer programming subproblem.

Ahuja et al. [1] consider a simplified version of the problem in which they assume that each train runs every day of the week (which thus reduces the weekly problem to an equivalent and simpler daily problem), ignore the train timings, and propose a time-space network. However, they do not solve the model directly and propose another heuristic method to solve the problem. The heuristic solves the problem in two phases. In the first phase, they determine the train network; i.e., train origins, destinations, and routes as well as block-to-train assignments, that takes into account the blocking network. In the second phase, train frequencies and timetables are determined for the train network.

Generally, train routing and train scheduling problems are solved separately since

they are hard problems to combine. However, there are also optimization models that integrate both routing and scheduling decisions into a single model. Morlok and Peterson [45] are two of the pioneers. The costs considered include train and crew costs, intermediate transfer point costs, and car-time costs. The model incorporates constraints on the maximum number of cars per train as well as scheduling constraints requiring some blocks to be delivered within certain time windows. The model was applied to a very small instance and solved with a branch-and-bound procedure.

Huntley et al. [29] introduces a computerized routing and scheduling system for CSX Transportation strategic planning. The output of the model is the sequence of train links that each block should follow from origin to destination, as well as the departure times for all train links. The problem is solved using simulated annealing and a perturbation operator that inserts or deletes a stop from the route of a block, and adjusts the departure times of the trains. The system was tested on a problem involving 166 blocks and 41 transfer points. Gorman [25] introduces a combination of genetic and tabu search algorithms to address weekly train routing and scheduling problems.

1.5 Thesis focus and organization

The primary contribution of this dissertation is the introduction of a modeling technique for pickup and delivery problems with time windows along with its extensions and its application in ship and train routing and scheduling problems.

This work is an extension to the work on the tramp problem studied by Jetlund and Karimi [32] which is an actual problem faced by a chemical shipping company in South-East Asia. The work on the train routing and scheduling problem is also inspired by a real problem faced by a leading railroad company in the United States.

Our novel formulation involves quite a number of bilinear constraints that are linearized using the scheme for bilinear terms introduced by Al-Khayyal and Falk [4]. It is a linearization scheme based on convex and concave envelopes which allows to

reformulate our model into an *equivalent* Mixed Integer Linear Program (MILP).

The remainder of this thesis is organized as follows: Chapters 2-5 discuss the ship routing and scheduling problem while Chapter 6 focuses on the train routing and scheduling problem. In Chapter 2 we give a detailed description of the ship routing and scheduling problem studied in this thesis, followed by the underlying assumptions and model formulation. We follow the notation of Jetlund and Karimi [32] to formulate our multi-ship problem. We also exploit the special structure embedded in our model and uncover the totally-unimodular structure enjoyed by some part of the activity matrix. Chapter 3 will show several attempts to solve the problem heuristically. Extensive computational results are presented to show how good the heuristics perform in solving randomly generated problems. We describe how these random test problems are generated. We also solve the random test problems to optimality to show the performance of the proposed heuristics, in terms of solution quality and computational time. We then introduce an upper-bounding problem to our model in Chapter 4. We also show some computational results on the performance of the upper-bounding problems. In Chapter 5, we will demonstrate the usefulness of our model to reflect problems with time aspects. From our ship problem, we relax the original (hard) pickup time window to soft time windows, allowing a cargo to be picked up outside its pickup time window by incurring some penalty cost. We also extend our model to accommodate cargo transfer between ships. In this case, we no longer require a cargo to be picked up and downloaded by the same ship. Instead, we allow at most two ships to serve one particular cargo. Of course we may need extra constraints in our model. We use quite a number of bilinear constraints that can be exactly linearized using a linearization scheme. We demonstrate the ideas by solving small test problems. We will also discuss the application of the model to solve train routing and scheduling problems faced by a real railroad company in Chapter 6. The complete train model and some computational results are presented. Chapter 7 will close our discussion by providing a summary of the thesis, suggested future work, and

some concluding remarks.

CHAPTER II

PROBLEM DESCRIPTION AND MODEL FORMULATION

The problem we consider to illustrate our model is a real planning problem faced by a major, multi-national shipping company that operates a fleet of multi-parcel chemical tankers in Asia. The carriers operate in Asia, but the fleet also serves Australia, India, and the Middle East. The fleet serves in redistributing cargos to respective download ports in these regions. The fleet also deals with freight contracts with manufacturers in the region or spot requests. The problem has been studied by Jetlund and Karimi [32] in which they suggest an initial version of the model studied in this thesis. Here we generalize the use of the model to serve most of the cargo types. There are time-window requirements for cargo pickup and download. The charterer would normally require a cargo to be picked up within a specific period. Typically, they demand the delivery to take place as soon as possible after pickup and the longest time they would tolerate is six weeks after the pickup happens. But generally, the ship notifies the customers of its estimated time of arrival at the download port and updates them with additional information if there are any changes to the ship's schedule.

2.1 Problem description

We are given a fleet of ships $\mathcal{S}$ with different characteristics, each of which can be owned by a different owner. They have different capacities $VMAX_s$, fuel consumption rates FC_s, and charter costs TCC_s. A ship's capacity is proportional to its size and can be measured in m^3, tonnes, number of compartments, or any other metric. Each ship also has its own operating rate, measured in tonnes of fuel or money spent per nautical mile. At time zero, we know the current locations of all ships as well as

their immediate destinations. Also, we know that each ship contains a set of onboard cargos $\mathcal{L}$ (loaded) to be delivered to their various destinations.

A ship $s \in \mathcal{S}$ can visit a collection of ports from the set $\mathcal{I}$. When a ship visits a port $i \in \mathcal{I}$, it has to pay a port charge PC_{is} per visit, which normally depends on the size or capacity of the ship. This port charge does not depend on how long the ship spends at that particular port. Typically, a ship anchors after arrival at a port and waits for an inspection to be done by port officers. After it passes the inspection procedure, then it may berth to unload and/or load cargos. Before it may leave the port to continue its voyage, it must again go through some inspections. We denote the *total inspection time* of any ship in any port as T_{adm}, which typically is the same for all ports.

Also, we are given the distance matrix between ports, a set of onboard cargos $\mathcal{L}$, and a set of unloaded cargos $\mathcal{U}$ at various ports. Each unloaded cargo $j \in \mathcal{U}$ has specified pickup port and destination port information, denoted by PP_j and DP_j, respectively. In addition, each cargo also has size information V_j which can be measured by its weight (tonnes) or volume (m^3), consistent with how the capacity of a ship is measured. A cargo also has hard time-window information during which it can be picked up from its originating port. This time-window, which is often referred to as *laycan*, consists of the earliest pickup time EPT_j and the latest pickup time LPT_j. Similarly, EDT_j and LDT_j denote the earliest and the latest download time windows, respectively. In a later extension, we will also introduce *soft* time-windows where a cargo can still be picked up outside its original time-window but incurs a penalty. A cargo is loaded with a loading rate LR_j which depends on the characteristics of the cargo itself. Similarly, we have a download rate DR_j for cargo J. Whenever cargo j is transferred from its pickup port to download port, a shipping revenue of SR_j is gained. It is important to mention that our fleet of ships do not have to serve all of the unloaded cargos in $\mathcal{U}$, thereby foregoing some potential revenue.

The objective is to select which cargos to be served by which ship and determine

the route that each ship should follow during the planning horizon, such that the total profit of all ships is maximized. The segment of a ship's journey from one port to the immediately following port is often referred to as a *leg*. The profit for each ship is the revenue from serving cargos minus fuel cost, all port costs, and a charter cost. The planning horizon is usually 3 to 4 weeks long, and the number of ports visited (equal to the number of legs) is predetermined by the schedule planner. The number of legs of one ship may differ from another, but every ship has to deliver at least all onboard cargos.

We assume that our schedule is a short-term plan that is to be revised frequently. We expect to run our problem over and over again once new information becomes available; for example, if there is bad weather, unexpected delays, more new cargos become available for pickup, or any other reasons that force us to reconfigure our current solution.

2.2 Assumptions

The following assumptions are made to simplify the notation or to estimate values of some parameters:

(i) Each ship $s \in \mathcal{S}$ belongs to a certain class based on its dead-weight capacity. The port cost PC_i is estimated as the average cost of ships in the respective class.

(ii) Ships sail at a constant average speed of v nautical miles per hour for all journeys.

(iii) Any ship may visit any port. In reality, one port can only serve certain kinds of ships. Hence, the compatibility between ships and ports is not considered.

(iv) Fuel consumption is a linear function of the sailed distance, independent of load or the draught of the ship. The draught of a ship is defined as the depth of a ship's keel below the surface, especially when loaded. The actual fuel

consumption depends on the draught of the ship as well as the sailing speed of the ship in a highly non-linear fashion.

(v) At time zero, for every ship we are given the initial destination and estimated time of arrival at the designated port. Each ship will complete its journey to the target port before a new schedule is created for it. In reality, a ship may change directions in the middle of a voyage.

(vi) A ship cannot visit a port more than once during the planning horizon. In most situations, a ship would visit a port only once in each planning horizon. However, our schedule is expected to be revised frequently, so in fact a ship may have multiple visits to the same port.

(vii) The inspection times before berthing and before leaving the port are both $0.5T_{adm}$. In other words, an equal amount of time is needed before and after for a total inspection time of T_{adm} per port visit.

(viii) In this high level planning, we do not consider the process of assigning cargos to compartments on the ship. We assume that a feasible stowage arrangement exists, as long as the capacity constraint holds.

(ix) Once the ship loads a cargo, it must deliver that cargo and cannot transfer the cargo to another ship. We will relax this assumption in one of our extensions which modifies the model to allow transfers between ships.

(x) Each cargo has its time-windows for pickup and delivery. Typically, delivery time-windows are not defined as one of the company's business rules. However, since the generated schedule is short-term and we require every picked-up cargo to be delivered within the time horizon, the time horizon itself can act as a deadline. In other words, by setting the time horizon as 4 weeks, we require each served cargo to be delivered within 4 weeks. Later we will also introduce

soft time-windows for cargo pickups where we allow some cargos to be picked-up outside their original time windows with some penalty.

(xi) Loading and downloading rates of each cargo j are given by the average of total cargo volume transferred divided by the relevant loading and downloading rates. It is assumed to be the same for all types of cargos. Depending on data availability, we can always differentiate the rates for each cargo. The loading and downloading rates for each cargo are conservative measures since we assume only one cargo can be loaded or discharged at a time. A ship sometimes would be able to load and/or discharge multiple cargos at the same time.

(xii) Pickup and delivery time-window requirements are measured by the arrival time of a ship at the respective ports, not the actual time the cargos are fully loaded or unloaded.

2.3 Model formulation

The problem was visited by Jetlund and Karimi [32] and is formulated as a Mixed Integer Linear Program (MILP). We follow their notation and investigate the problem for the multi-ship case. We model the schedule for a ship s as a series of (K_s+1) sailing legs, where K_s is the number of sailing legs a planner schedules. Sailing leg $k = 0$ is the leg where a ship reaches its destination port at time zero. This initial target port information is given prior to the new schedule generation. Let $\mathcal{K}_s = \{0, 1, ..., K_s\}$ be the set of legs for ship s. In reality, with the emerging technology of Global Positioning Systems (GPS) we may model the initial ship location as an imaginary port allowing the ship to change its first destination at time zero.

In addition to real ports, a dummy port $i = 0$ is introduced with zero port cost and zero distance from any other port. There is no cargo originating from and to be discharged at this dummy port. Therefore the model also forces a condition that after a ship visits this dummy port, it will have to stay there until the end of the

planning horizon. We recognize this dummy port as an absorbing port used to idle any ship, if this option is considered profitable.

2.3.1 Routing constraints

The routing constraints define and link the sequence of arrivals and departures of the various ships to and from available ports. We define binary variables X_{isk_s} to model whether or not ship s is at port i at the end of sailing leg k_s. Note that the index k depends on s, because for different ships we may have different numbers of legs ($k_s \in \mathcal{K}_s$), where $\mathcal{K}_s$ denotes the set of legs for ship s. But for simplicity, let us drop the subscript s from index k_s. We define

$$X_{isk} = \begin{cases} 1 & \text{if ship } s \text{ reaches port } i \text{ at the end of leg } k \\ 0 & \text{otherwise} \end{cases}$$

and let T_{sk} denote the time at which ship s arrives at any port at the end of leg k.

Visit exactly one port in each leg

A ship can only visit one port during each leg. It can be any of the service ports or the dummy port

$$\sum_i X_{isk} = 1, \qquad s \in \mathcal{S}, k \in \mathcal{K}_s. \tag{1}$$

Maximum one visit for all ports $i \in \mathcal{I} \setminus \{0\}$

From assumption (vi) of section 2.2, a ship can visit a specific port at most once during each leg, except for the dummy port, so

$$\sum_k X_{isk} \leq 1, \qquad i \in \mathcal{I} \setminus \{0\}, s \in \mathcal{S}. \tag{2}$$

Dummy port as an absorbing port

To prevent a ship from transiting through the dummy port, we force that once a ship visits the dummy port, it has to stay there until the end of the planning horizon. In other words, the dummy port is only visited by ships that are idle until the end of the planning horizon. We express this in the following constraint

$$X_{0sk} \leq X_{0s(k+1)}, \qquad s \in \mathcal{S}, k \in \mathcal{K}_s \backslash \{K_s\}. \tag{3}$$

Port-to-port travel

To model travel from the current port to the next port, binary transition variables are defined as

$$Z_{ilsk} = \begin{cases} 1 & \text{if ship } s \text{ visits port } i \text{ at the end of leg } k \\ & \text{and port } l \text{ at the end of leg } (k+1) \\ 0 & \text{otherwise.} \end{cases}$$

Clearly, $Z_{ilsk} = X_{isk} X_{ls(k+1)}$, for all $i, j \in \mathcal{I}, s \in \mathcal{S}, k \in \mathcal{K}_s \backslash \{K_s\}$. This represents a nonlinear relationship, but we utilize an equivalent linear reformulation of this type of constraint given by these two constraints

$$\sum_{l \in \mathcal{I}, l \neq i} Z_{ilsk} = X_{isk}, \qquad i \in \mathcal{I}, s \in \mathcal{S}, k \in \mathcal{K}_s \backslash \{K_s\}, \tag{4}$$

$$\sum_{i \in \mathcal{I}, i \neq l} Z_{ilsk_s} = X_{ls(k_s+1)}, \qquad l \in \mathcal{I}, s \in \mathcal{S}, k \in \mathcal{K}_s \backslash \{K_s\}. \tag{5}$$

By this equivalent linear formulation, we may treat Z_{ilsk} as a $0-1$ *continuous variable.*

2.3.2 Cargo movement constraints

We define binary variables Y_{js} to indicate whether a ship s should serve a cargo j

$$Y_{js} = \begin{cases} 1 & \text{if ship } s \text{ serves cargo } j \\ 0 & \text{otherwise.} \end{cases}$$

Clearly, $Y_{js} = 1$ for all onboard cargos $j \in \mathcal{L}$. In our current scenario, once a ship picks up a cargo from its pickup port, it has to carry it all the way to the destination port within the time horizon. To capture these movements, we define three types of binary variables XP_{jsk}, XD_{jsk}, and XC_{jsk} that later turn out can be set as $0-1$ continuous variables.

$$XP_{jsk} = \begin{cases} 1 & \text{if ship } s \text{ picks up cargo } j \text{ at the end of leg } k \\ 0 & \text{otherwise} \end{cases}$$

$$XD_{jsk} = \begin{cases} 1 & \text{if ship } s \text{ downloads cargo } j \text{ at the end of leg } k \\ 0 & \text{otherwise} \end{cases}$$

$$XC_{jsk} = \begin{cases} 1 & \text{if ship } s \text{ carries cargo } j \text{ onboard during leg } k \\ 0 & \text{otherwise} \end{cases}$$

Note that for all cargos $j \in \mathcal{L}$, $XC_{js0} = 1$ for their respective ship s.

Cargo pickups and downloads

For a ship s to load a cargo j at the end of leg k, the ship must visit the loading port PP_j at time T_{sk}, and of course it must service cargo j. Hence, we must have

$$XP_{jsk} = Y_j X_{(PP_j)sk}, \qquad j \in \mathcal{U}, s \in \mathcal{S}, k \in \mathcal{K}_s.$$

Similarly, for a ship s to unload a cargo j at the end of leg k, the ship must visit the unloading port DP_j at time T_{sk}, and it must service cargo j. Therefore,

$$XD_{jsk} = Y_{js} X_{(DP_j)sk}, \qquad j \in \mathcal{L} \cup \mathcal{U}, s \in \mathcal{S}, k \in \mathcal{K}_s.$$

These non-linear constraints are replaced by the following linear constraints. Firstly, when a ship s does service a cargo j (i.e., $Y_{js} = 1$), then it must be loaded in exactly one leg. Otherwise, when it does not serve the cargo, then it cannot be loaded in any leg. The same is true for downloads. Thus,

$$\sum_k XP_{jsk} = Y_{js}, \qquad j \in \mathcal{U}, s \in \mathcal{S}, \tag{6}$$

$$\sum_k XD_{jsk} = Y_{js}, \qquad j \in \mathcal{L} \cup \mathcal{U}, s \in \mathcal{S}. \tag{7}$$

Secondly, the pickup and download can happen only at a cargo's pickup and discharge ports, respectively. That is,

$$XP_{jsk} \leq X_{(PP_j)sk}, \qquad j \in \mathcal{U}, s \in \mathcal{S}, k \in \mathcal{K}_s, \tag{8}$$

$$XD_{jsk} \leq X_{(DP_j)sk}, \qquad j \in \mathcal{L} \cup \mathcal{U}, s \in \mathcal{S}, k \in \mathcal{K}_s. \tag{9}$$

By these constraints we are assured that we can treat variables XP_{jsk} and XD_{jsk} as 0-1 *continuous variables*.

Pickup before download

If a ship services a cargo, then it must visit the pickup port before it visits the delivery port, which is expressed in the following constraint

$$\sum_{k>0} k(XD_{jsk} - XP_{jsk}) \geq Y_{js}, \qquad j \in \mathcal{U}, s \in \mathcal{S}. \tag{10}$$

Carrying cargo from pickup to download port

Also, to make a ship carry a cargo j from its pickup port to its delivery port, the following constraint is used

$$XC_{js(k+1)} = XC_{jsk} + XP_{jsk} - XD_{jsk}, \qquad j \in \mathcal{L} \cup \mathcal{U}, s \in \mathcal{S}, k \in \mathcal{K}_s \backslash \{K_s\}. \quad (11)$$

Since variables XP_{jsk} and XD_{jsk} are automatically set to be binary, the same applies to variable XC_{jsk}. Clearly, $XC_{js0} = 1$ for all cargos $j \in \mathcal{L}$ carried by ship s.

Ship capacity

Every time a ship carries some cargos onboard, they must be within the carrying capacity of the ship. Thus,

$$\sum_j V_j XC_{jsk} \leq VMAX_s, \qquad s \in \mathcal{S}, k \in \mathcal{K}_s. \quad (12)$$

One cargo is served by one ship

To ensure that a cargo is serviced by at most one ship, the following constraint is imposed

$$\sum_s Y_{js} \leq 1, \qquad j \in \mathcal{L} \cup \mathcal{U}. \quad (13)$$

2.3.3 Time constraints

We let T_{sk} denote the time at which a ship s arrives at any port at the end of its leg k. After a ship performs its service at a port, it will continue its journey to the next port. Let TT_{sk} denote the time a ship s takes to travel from a current port at the end of leg k to the next port at the end of leg $(k+1)$. If the distance between port i and port l is Dis_{il}, then TT_{sk} can be expressed as

$$TT_{sk} = \sum_i \sum_l \frac{\text{Dis}_{il} Z_{ilsk}}{24 v_s} \qquad s \in \mathcal{S}, k \in \mathcal{K}_s \backslash \{K_s\}. \quad (14)$$

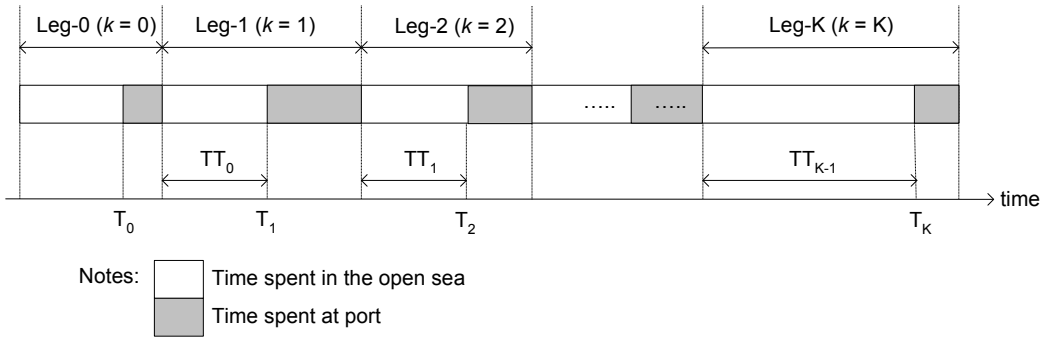

Figure 1: Time representation with port arrivals at T_{sk} and time travel between ports TT_{sk}.

Here, v_s denotes the speed of ship s in nautical miles per hour. Notice that only one Z_{ilsk} can equal 1 for all i and all j. Hence, TT_{sk} is actually the distance between two ports visited at the ends of legs k and $(k+1)$ divided by the speed of the ship. Figure 1 shows the connection between T_{sk} and TT_{sk} in a slot-based time representation.

Pickup and download time windows

We must ensure that whenever a ship serves a cargo, it has to pickup the cargo within the pickup laycan. First, a pickup can only happen before the latest pickup time expires. Assuming that a ship requires half of the total administrative time before it can actually perform a pickup in any port, the following constraint is defined:

$$T_{sk} \leq (LPT_j - \frac{1}{2}T_{adm})XP_{jsk} + M(1 - XP_{jsk}), \qquad j \in \mathcal{U}, s \in \mathcal{S}, k \in \mathcal{K}_s \backslash \{K_s\}. \quad (15)$$

The constant M in the above constraint is a large number that functions when a ship does not serve the cargo. And second, the arrival time at the next port has to meet the pickup time windows for whatever cargos that are being picked up at the next port. Therefore,

$$T_{s(k+1)} \geq (EPT_j + \frac{1}{2}T_{adm})XP_{jsk} + \frac{V_j XP_{jsk}}{LR_j} + TT_{sk}, \qquad j \in \mathcal{U}, s \in \mathcal{S}, k \in \mathcal{K}_s \backslash \{K_s\}. \tag{16}$$

Similarly, an unloading has to be done within the download time-windows. Therefore,

$$T_{sk} \leq (LDT_j - \frac{1}{2}T_{adm})XD_{jsk} + M(1 - XD_{jsk}), \qquad j \in \mathcal{L} \cup \mathcal{U}, s \in \mathcal{S}, k \in \mathcal{K}_s, \tag{17}$$

$$T_{s(k+1)} \geq (EDT_j + \frac{1}{2}T_{adm})XD_{jsk} + \frac{V_j XD_{jsk}}{DR_j} + TT_{sk}, \qquad j \in \mathcal{L} \cup \mathcal{U}, s \in \mathcal{S}, k \in \mathcal{K}_s \backslash \{K_s\}. \tag{18}$$

Time spent at a port

Since we assume that a ship can load or unload one cargo at a time, we also have to consider the total time required to serve all cargos. Total service time in a port must exceed the time required for inspections, plus the time for discharging all delivery cargos, plus the time for loading pickup cargos. Also, particularly for the dummy port, we do not have administrative time. Hence,

$$T_{s(k+1)} \geq T_{sk} + T_{adm}(1 - X_{0sk}) + \sum_j \frac{V_j XD_{jsk}}{DR_j} + \sum_j \frac{V_j XP_{jsk}}{LR_j} + TT_{sk}, s \in \mathcal{S}, k \in \mathcal{K}_s \backslash \{K_s\}. \tag{19}$$

2.3.4 Objective function

The objective is to maximize profits, i.e. revenues generated from all serviced cargos, less all fuel costs FC_s, port charges PC_{is}, and time-charter costs TCC_s.

$$(OBJ)\ \text{Maximize} \begin{cases} \sum_j \sum_s SR_j Y_{js} & \text{Revenue} \\ -\sum_s \sum_k FC_s v_s TT_{sk} & \text{- Fuel Costs} \\ -\sum_s \sum_i \sum_k PC_{is} X_{isk} & \text{- Port Costs} \\ -\sum_s TCC_s \times \left(T_{K_s,s} + \sum_j \frac{V_j XD_{jsK_s}}{DR_j} \right) & \text{- Time Charter Costs} \end{cases} \tag{20}$$

where $j \in \mathcal{L} \cup \mathcal{U}, i \in \mathcal{I}, s \in \mathcal{S}$, and $k \in \mathcal{K}_s$. This completes the formulation for our multi-ship scheduling problem.

To summarize, we have the complete problem modeled as a MILP problem. For the sake of completeness we derive the dual form of the MILP relaxation which can be found in Appendix B.

2.3.5 Complete multi-ship problem

$$(OBJ)\ \text{Maximize} \begin{cases} \sum_j \sum_s SR_j Y_{js} & \text{Revenue} \\ -\sum_s \sum_k FC_s v_s TT_{sk} - \sum_s FC_s v_s T_{s0} & \text{- Fuel Costs} \\ -\sum_s \sum_i \sum_k PC_{is} X_{isk} & \text{- Port Costs} \\ -\sum_s TCC_s \times \left(T_{K_s,s} + \sum_j \frac{V_j XD_{jsK_s}}{DR_j} \right) & \text{- Time Charter Costs} \end{cases}$$

$$\begin{array}{lrcll}
(A_{sk}) & \sum_i X_{isk} & = & 1 & s \in \mathcal{S}, k \in \mathcal{K}_s \\
(B_{is}) & \sum_k X_{isk} & \leq & 1 & i \in \mathcal{I} \setminus \{0\}, s \in \mathcal{S} \\
(C_{sk}) & X_{0sk} - X_{0s(k+1)} & \leq & 0 & s \in \mathcal{S}, k \in \mathcal{K}_s \backslash \{K_s\} \\
(D_{isk}) & \sum_l Z_{ilsk} - X_{isk} & = & 0 & i \in \mathcal{I}, s \in \mathcal{S}, k \in \mathcal{K}_s \backslash \{K_s\} \\
(E_{isk}) & \sum_l Z_{lisk} - X_{is(k+1)} & = & 0 & i \in \mathcal{I}, s \in \mathcal{S}, k \in \mathcal{K}_s \backslash \{K_s\} \\
(F_{js}) & \sum_k XP_{jsk} - Y_{js} & = & 0 & j \in \mathcal{U}, s \in \mathcal{S} \\
(G_{jsk}) & XP_{jsk} - X_{(PP_j)sk} & \leq & 0 & j \in \mathcal{U}, s \in \mathcal{S}, k \in \mathcal{K}_s \\
(H_{js}) & \sum_k XD_{jsk} - Y_{js} & = & 0 & j \in \mathcal{L} \cup \mathcal{U}, s \in \mathcal{S} \\
(I_{jsk}) & XD_{jsk} - X_{(DP_j)sk} & = & 0 & j \in \mathcal{L} \cup \mathcal{U}, s \in \mathcal{S}, k \in \mathcal{K}_s \\
(J_{js}) & \sum_{k>0} k(XP_{jsk} - XD_{jsk}) + Y_j & \leq & 0 & j \in \mathcal{U}, s \in \mathcal{S} \\
(KD_{jsk}) & -XC_{js(k+1)} + XC_{jsk} + XP_{jsk} - XD_{jsk} & = & 0 & j \in \mathcal{L} \cup \mathcal{U}, s \in \mathcal{S}, k \in \mathcal{K}_s \backslash \{K_s\}
\end{array}$$

$$
\begin{array}{lrcll}
(LD_{sk}) & \sum_j V_j Y_{jsk} & \leq & VMAX_s & s \in \mathcal{S}, k \in \mathcal{K}_s \\
(MD_{sk}) & TT_{sk} - \sum_i \sum_l \frac{\text{Dis}_{il} Z_{ilsk}}{24 v_s} & = & 0 & s \in \mathcal{S}, k \in \mathcal{K}_s \\
(N_{jsk}) & T_{sk} + (M - LPT_j + \frac{1}{2} T_{adm}) XP_{jsk} & \leq & M & j \in \mathcal{U}, s \in \mathcal{S}, k \in \mathcal{K}_s \backslash \{K_s\} \\
(O_{jsk}) & -T_{s(k+1)} + (EPT_j + \frac{1}{2} T_{adm}) XP_{jsk} + ... & & & \\
 & \frac{V_j XP_{jsk}}{LR_j} + TT_{sk} & \leq & 0 & j \in \mathcal{U}, s \in \mathcal{S}, k \in \mathcal{K}_s \backslash \{K_s\} \\
(P_{sk}) & -T_{s(k+1)} + T_{sk} - T_{adm} X_{0sk} + ... & & & \\
 & \sum_j \frac{V_j XD_{jsk}}{DR_j} + \sum_j \frac{V_j XP_{jsk}}{LR_j} + TT_{sk} & \leq & -T_{adm} & s \in \mathcal{S}, k \in \mathcal{K}_s \backslash \{K_s\} \\
(Q_j) & \sum_s Y_{js} & \leq & 1 & j \in \mathcal{U} \\
(R_{jsk}) & T_{sk} + (M - LDT_j + \frac{1}{2} T_{adm}) XD_{jsk} & \leq & M & j \in \mathcal{U}, s \in \mathcal{S}, k \in \mathcal{K}_s \backslash \{K_s\} \\
(S_{jsk}) & -T_{s(k+1)} + (EDT_j + \frac{1}{2} T_{adm}) XD_{jsk} + ... & & & \\
 & \frac{V_j XD_{jsk}}{DR_j} + TT_{sk} & \leq & 0 & j \in \mathcal{U}, s \in \mathcal{S}, k \in \mathcal{K}_s \\
(VX_{isk}) & X_{isk} & \leq & 1 & i \in \mathcal{I}, s \in \mathcal{S}, k \in \mathcal{K}_s \\
(VZ_{ilsk}) & Z_{ilsk} & \leq & 1 & i, l \in \mathcal{I}, s \in \mathcal{S}, k \in \mathcal{K}_s \backslash \{K_s\} \\
(VY_{js}) & Y_{js} & \leq & 1 & j \in \mathcal{L} \cup \mathcal{U}, s \in \mathcal{S} \\
(VXP_{jsk}) & XP_{jsk} & \leq & 1 & j \in \mathcal{U}, s \in \mathcal{S}, k \in \mathcal{K}_s \\
(VXD_{jsk}) & XD_{jsk} & \leq & 1 & j \in \mathcal{L} \cup \mathcal{U}, s \in \mathcal{S}, k \in \mathcal{K}_s \\
(VXC_{jsk}) & XC_{jsk} & \leq & 1 & j \in \mathcal{L} \cup \mathcal{U}, s \in \mathcal{S}, k \in \mathcal{K}_s
\end{array}
$$

2.4 Example

Our example is taken from a real chemical shipping company which operates a fleet of small, multi-parcel ships in the Asia Pacific region. At one time in the past (time zero is 17 April, 12:00AM) the company had 10 ships and 79 cargo requests involving 36 ports. Among these 79 cargos, at time zero, 37 of them were loaded on their corresponding ships while 42 others were new potential cargos. This cargo information is shown in Tables 1 and 2. We also have information about each ship and port as shown in Tables 3 and 4, respectively.

We first repeat the work done by Jetlund and Karimi [32] to solve several one-ship problems, meaning that we ignore the existence of all but one ship and solve the problem. The problems are solved using by CPLEX 9.000 on a four-CPU Sun E450 server machine under all of the default options of CPLEX for each one-ship problem.

Table 1: Unloaded cargo information details

Cargo	Origin	Destination	Pickup Time Windows	Latest Download	Volume (tonnes)	Shipping Rate (USD)
1	Karimun	Shuidong	25-29 April	10 June	950	45,125
2	Karimun	Shekou	25-29 April	10 June	596	29,800
3	Karimun	Taichung	25-29 April	10 June	1049	31,470
4	Karimun	Kaohsiung	25-29 April	10 June	700	28,000
5	Karimun	Shanghai	25-29 April	10 June	501	20,040
6	Karimun	Bangkok	25-29 April	10 June	2092	62,760
7	Karimun	Jasaan	25-29 April	10 June	1000	45,000
8	Karimun	Anyer	25-29 April	10 June	1011	28,308
9	Karimun	Ningbo	23-27 April	8 June	1400	50,400
10	Karimun	Port Kelang	23-27 April	8 June	500	30,000
11	Ulsan	Ningbo	26-29 April	10 June	995	33,830
12	Ulsan	Ningbo	26-29 April	10 June	678	23,052
13	Ulsan	Shanghai	26-29 April	10 June	1000	34,000
14	Ulsan	Shanghai	26-29 April	10 June	505	17,170
15	Ulsan	Nantong	26-29 April	10 June	1000	36,000
16	Ulsan	Nantong	26-29 April	10 June	315	11,340
17	Singapore	Shekou	21-25 April	6 June	2700	67,500
18	Singapore	Shekou	21-25 April	6 June	350	8,750
19	Singapore	Shekou	21-25 April	6 June	600	15,000
20	Singapore	Botany Bay	14-23 April	4 June	800	44,800
21	Kuantan	Shanghai	22-27 April	8 June	800	30,000
22	Kuantan	Shanghai	22-27 April	8 June	300	11,250
23	Kuantan	Ulsan	22-27 April	8 June	400	15,000
24	Kuantan	Ulsan	22-27 April	8 June	700	26,250
25	Kuantan	Shanghai	22-27 April	8 June	1000	37,500
26	Singapore	Kandla	18-25 April	6 June	1000	26,250
27	Singapore	Kandla	18-25 April	6 June	500	13,125
28	Singapore	Kandla	18-25 April	6 June	500	13,125
29	Singapore	Kandla	18-25 April	6 June	300	7,875
30	Karimun	Maptaphut	25-29 April	10 June	500	16,000
31	Karimun	Maptaphut	25-29 April	10 June	1000	32,000
32	Karimun	Bangkok	25-29 April	10 June	250	8,000
33	Singapore	Kerteh	25-30 April	11 June	350	23,999.5
34	Kuantan	Bangkok	01-05 May	16 June	500	20,000
35	Kuantan	Bangkok	01-05 May	16 June	200	8,000
36	Onsan	Paradip	21-25 April	6 June	6000	289,800
37	Ulsan	Lanshantao	21-25 April	6 June	300	9,000
38	Ulsan	Shanghai	21-25 April	6 June	600	18,000
39	Brisbane	Kaohsiung	24-28 April	9 June	1100	57,607
40	Brisbane	Shanghai	24-28 April	9 June	2700	141,399
41	Brisbane	Zhapu	24-28 April	9 June	4500	235,665
42	Brisbane	Taichung	24-28 April	9 June	150	7,855.5

Table 2: Loaded cargo information details

Cargo	Origin	Destination	Latest Download	Volume (tonnes)	Ship Rate (USD)	Status at time zero
43	Ulsan	Bangkok	17 May	315	12,600	Loaded on ship 1
44	Ulsan	Bangkok	17 May	315	12,600	Loaded on ship 1
45	Ulsan	Bangkok	17 May	315	12,600	Loaded on ship 1
46	Ulsan	Bangkok	17 May	199	7,960	Loaded on ship 1
47	Ulsan	Kuantan	17 May	1490	48,425	Loaded on ship 1
48	Ulsan	Singapore	17 May	455	13,650	Loaded on ship 1
49	Ulsan	Singapore	17 May	105	3,150	Loaded on ship 1
50	Ulsan	Singapore	17 May	509	15,270	Loaded on ship 1
51	Ulsan	Tanjung Priok	17 May	210	15,006.6	Loaded on ship 1
52	Ulsan	Tanjung Priok	17 May	210	15,006.6	Loaded on ship 1
53	Kuantan	Yosu	17 May	850	36,125	Loaded on ship 2
54	Kuantan	Ulsan	17 May	300	12,000	Loaded on ship 2
55	Kuantan	Ulsan	17 May	300	12,000	Loaded on ship 2
56	Abu Jubail	Tanjung Priok	17 May	2086	74,053	Loaded on ship 3
57	Ulsan	Karimun	17 May	3000	120,000	Loaded on ship 4
58	Ulsan	Wellington	17 May	500	30,750	Loaded on ship 5
59	Ulsan	Timaru	17 May	500	30,750	Loaded on ship 5
60	Ulsan	New Plymouth	17 May	50	3,075	Loaded on ship 5
61	Ulsan	Auckland	17 May	50	3,075	Loaded on ship 5
62	Yosu	Yingkou	17 May	5359	96,462	Loaded on ship 6
63	Taichung	Nantong	17 May	1001	36,036	Loaded on ship 7
64	Taichung	Ningbo	17 May	650	26,000	Loaded on ship 7
65	Taichung	Jiangyin	17 May	1050	42,000	Loaded on ship 7
66	Mailiao	Shanghai	17 May	2000	52,000	Loaded on ship 7
67	Mailiao	Shanghai	17 May	500	13,000	Loaded on ship 7
68	Karimun	Davao	17 May	800	60,000	Loaded on ship 9
69	Karimun	Batangas	17 May	350	29,998.5	Loaded on ship 9
70	Kuantan	Batangas	17 May	300	24,999	Loaded on ship 9
71	Kerteh	Batangas	17 May	500	10,800	Loaded on ship 9
72	Kerteh	Mailiao	17 May	2000	43,200	Loaded on ship 9
73	Kerteh	Kaohsiung	17 May	1300	28,080	Loaded on ship 9
74	Kuantan	Batangas	17 May	300	24,999	Loaded on ship 9
75	Karimun	Shuidong	17 May	731	31,067.5	Loaded on ship 10
76	Karimun	Shuidong	17 May	488	20,740	Loaded on ship 10
77	Karimun	Xiaohudao	17 May	1000	40,000	Loaded on ship 10
78	Karimun	Xiaohudao	17 May	1000	26,000	Loaded on ship 10
79	Karimun	Xiaohudao	17 May	850	22,100	Loaded on ship 10

Table 3: Ship information details

Ship	Size (dwt)	Cost (USD/day)	Immediate Destination	ETA (day)	Fuel Cost (USD/nm)	Consumption (tonnes/day at 13 knots)
1	11000	9000	Bangkok	1.620	7.18	14
2	11000	9000	Yosu	1.875	7.18	14
3	11000	8000	Tanjong Priok	0.083	7.18	14
4	8200	8000	Singapore	3.625	6.15	12
5	8200	7000	Wellington	1.573	6.15	12
6	5800	7000	Yingkou	1.323	6.15	12
7	5800	7000	Ningbo	2.865	6.15	12
8	5800	7000	Ulsan	3.750	6.15	12
9	5800	7000	Davao	2.031	6.15	12
10	6000	7000	Shuidong	0.367	6.15	12

The ten problems were solved to optimality within a time range between 5.18 and 240.53 seconds. Each of our one-ship problems on average has 338 integer and 11,256 continuous variables, and 12,700 constraints. The result of solving all 10 one-ship problems is shown in Table 5 and Figure 2.

Our one-ship solution does not provide a feasible solution to our multi-ship problem. Since we only consider one ship at a time, a single cargo may be served by multiple ships, especially if the profit generated by serving that cargo is attractive.

Remark 2.4.1

(i) The optimal result from many one-ship problems are used to generate some heuristics for solving the multi-ship problem. These are presented in Chapter 3.

(ii) Our 10-ship problem involves 79 cargos (loaded and unloaded) and 37 ports. The number of integer and continuous variables are 3,380 and 112,560, respectively. The total number of constraints is 127,079. It is virtually impossible to solve such a multi-ship problem using the default options of CPLEX within a reasonable amount of time; i.e., the amount of time a ship would tolerate waiting for its schedule to be constructed.

Table 4: Port information details

Port Number	Port Name	Port Cost (USD) 9000-11000 dwt	Port Cost (USD) 6000-9000 dwt
1	Anyer	6250	4853
2	Auckland	5000	4500
3	Bangkok	6048	4594
4	Batangas	9370	8230
5	Botany Bay	10725	9000
6	Brisbane	6846	5500
7	Davao	7421	7085
8	Jasaan	7500	7000
9	Jiangyin	5224	4599
10	Kandla	5845	5500
11	Kaohsiung	4644	4188
12	Karimun	1988	1629
13	Kerteh	10957	9361
14	Kuantan	3608	3210
15	Lanshantao	5312	4127
16	Maptaphut	5681	4819
17	Mailiao	5900	5112
18	Nantong	6302	4543
19	New Plymouth	4839	4325
20	Ningbo	4740	3116
21	Onsan	5587	3688
22	Paradip	6501	5764
23	Port Kelang	3686	2513
24	Shanghai	6177	5125
25	Shekou	5757	4840
26	Shuidong	5862	5202
27	Singapore	7248	5348
28	Taichung	4624	4219
29	Tanjung Priok	5004	4724
30	Timaru	3748	3445
31	Ulsan	6591	5692
32	Wellington	4044	4000
33	Xiaohudao	7809	6760
34	Yingkou	5000	4325
35	Yosu	5772	5056
36	Zhapu	5000	4000

Table 5: Optimal one-ship solutions for all ten ships

Ship	Size (dwt)	Cost ($/day)	Immediate Destination	Arrival (days)	Cargos onboard	New cargos	Ship Profit (USD)
1	11,000	9,000	Bangkok	1.620	43-52	2,5,9,17-19,21-22,25	158,412
2	11,000	9,000	Yosu	1.875	53-55	11-15, 36, 38	191,288
3	11,000	8,000	Tj. Priok	0.083	56	1-2, 6, 17-19, 30-32	144,742
4	8,200	8,000	Singapore	3.625	57	1-2, 5, 9, 17, 21, 25	195,108
5	8,200	7,000	Wellington	1.573	58-61	40-41	171,609
6	5,800	7,000	Yingkou	1.323	62	11-16, 38	138,912
7	5,800	7,000	Ningbo	2.865	63-67	13-14, 38	115,221
8	5,800	7,000	Ulsan	3.750	-	11-16, 38	46,841.2
9	5,800	7,000	Davao	2.031	68-74	11-14	160,653
10	6,000	7,000	Shuidong	0.367	75-79	36	215,735

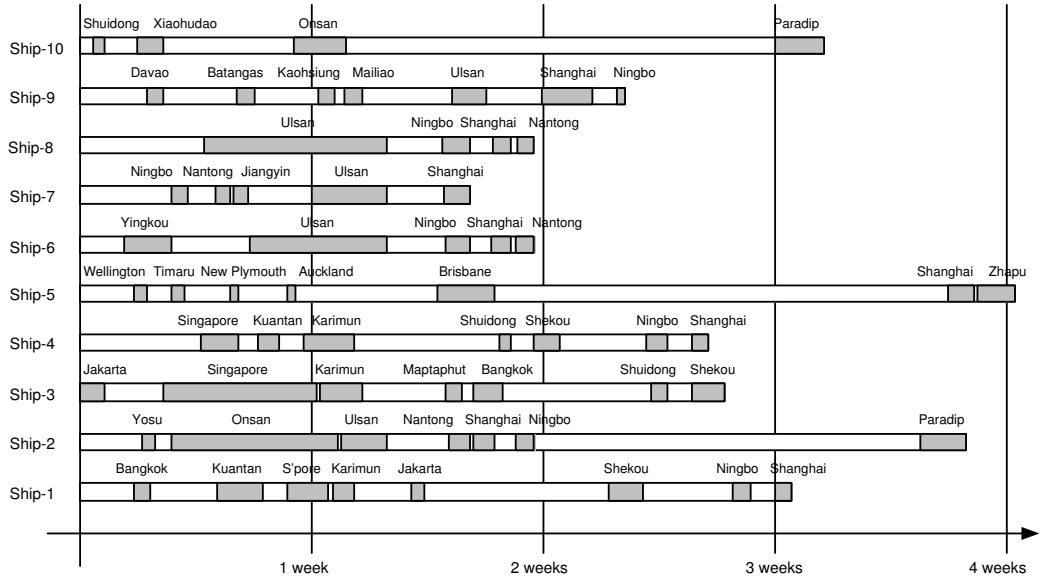

Figure 2: Gantt-chart for the optimal one-ship schedules

Table 6: Optimal 2-ship solutions for ships 8 and 9

Ship	Size (dwt)	Cost ($/day)	Immediate (Destination)	Arrival (days)	Cargos onboard	New cargos	Ship Profit (USD)
8	5,800	7,000	Ulsan	3.750	-	11-16, 38	46,841.2
9	5,800	7,000	Davao	2.031	68-74	1, 2, 4	114,589.8
Total					68-74	1, 2, 4, 11-16, 38	161,431

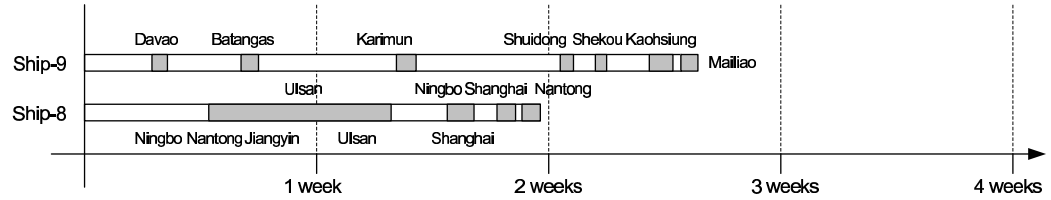

Figure 3: Gantt-chart for the optimal 2-ship schedules

We illustrate our multi-ship model by solving a 2-ship problem, consisting of ships 8 and 9. We ignore the rest of the ships and solve the problem of determining the optimal schedule only of ships 8 and 9. Using the default choices of CPLEX, the problem is solvable in 2469.24 seconds. In the optimal solution of the two one-ship problems, ships 8 and 9 are both competing to serve cargos 11-14. But in the optimal 2-ship schedule, cargos 11-14 are served by ship 8 with additional cargos 15, 16 and 38; while ship 9 is serving new cargos 1,2, and 4, that are not even carried in the one-ship optimal schedule. The result of this 2-ship problem is shown in Table 6 and Figure 3.

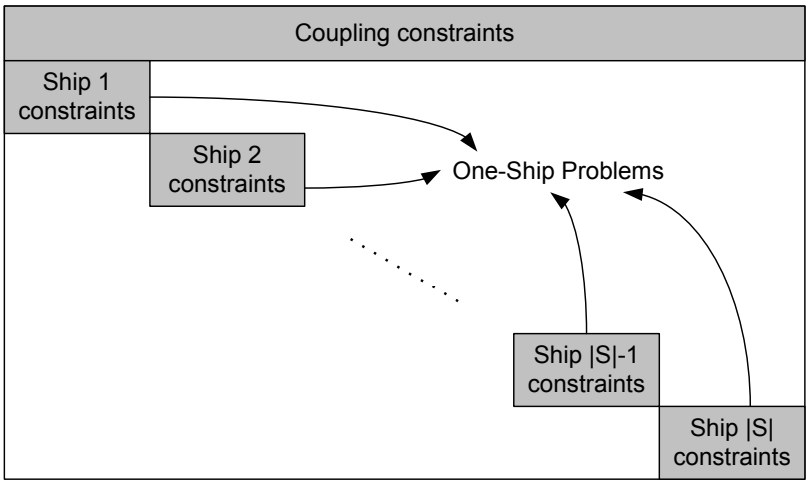

Figure 4: Constraint (13) acts as the coupling constraint, the remaining constraints can be decoupled into $|\mathcal{S}|$ smaller problems.

2.5 Decomposition

Decomposition is essential in order to solve MIP's of the size encountered in our model. Moreover, it is very natural to decompose the problem by ship. In this section we will show how to decompose the problem into smaller problems and exploit the special structure of the system.

Notice that constraint (13) can be viewed as a system of coupling constraint. The remaining other constraints can be decoupled into $|\mathcal{S}|$ smaller problems, as illustrated in Figure 4.

From the description of each variable we presented earlier, we have three kinds of variables; namely, variables related to ship movements (X_{isk}, Z_{ilsk}), variables related to cargo movements (Y_{js}, XP_{jsk}, XD_{jsk}, XC_{jsk}), and variables related to time constraints (T_{sk}, TT_{sk}). We use this classification as a basis for grouping our constraints.

Constraints (1), (2), (3), (4) and (5) only involve variables X_{isk} and Z_{ilsk}, so we group them into one block of constraints. Constraints (6), (7), and (11) involve cargo movements variables Y_{js}, XP_{jsk}, XD_{jsk}, and XC_{jsk} and we categorized them into one block of constraints. The third block consists of time constraints (14), (15), (16), and (19). Finally, we group the rest of the constraints (8), (9), (10), and (12) into

the fourth block. Hence our problem can be written as follow

$$\begin{array}{rrcrcrcl} \text{Maximize} & c_x^1\ x_1 & + & c_x^2\ x_2 & + & c_y\ y & & \\ \text{s.t.} & A_x^1\ x_1 & + & A_x^2\ x_2 & + & A_y\ y & \leq & a \\ & D_x^1\ x_1 & + & D_x^2\ x_2 & & & \leq & d \\ & E\ x_1 & & & & & \leq & e \\ & & & F\ x_2 & & & \leq & f \\ & & & x_1, x_2 \in \{0,1\},\ y \geq 0 & & & & \end{array} \tag{21}$$

where (c_x^1, c_x^2, c_y) is the objective coefficient row vector, A_x^1, A_x^2, A_y, D_x^1, D_x^2, E and F are real matrices, (a^T, d^T, e^T, f^T) is the right hand side column vector, and x_1, x_2 and y are column vectors, where

$$\begin{aligned} x &:= \langle X_{isk}, Z_{ilsk}, Y_{js}, XP_{jsk}, XD_{jsk}, XC_{jsk} \rangle \\ y &:= \langle TT_{sk}, T_{sk} \rangle \end{aligned}$$

range over $j \in \mathcal{L} \cup \mathcal{U},\ i, l \in \mathcal{I},\ s \in \mathcal{S},\ k \in \mathcal{K}_s$. We use the notation $\langle \cdot \rangle$ to denote the column vector with components ranging through the specified indices. Note that the variables X_{isk} and Y_{js} are pure binary variables and the remaining variables in (x_1,x_2) are forced to be binary by these two pure binary variables. The components of y are continuous. We further observe that matrices E and F are totally-unimodular, and we use this special property later. Also note that by using the relation reflected in equation (14) we can eliminate variable TT_{sk} from the system for all $s \in \mathcal{S}, k \in K_s$.

2.5.1 Total-unimodularity property

While the formulation does not follow a simple network structure, some constraints do form a totally-unimodular activity matrix.

Lemma 2.5.1 *For each ship $s \in \mathcal{S}$, the submatrices E and F as defined in the one-ship problem are totally-unimodular.*

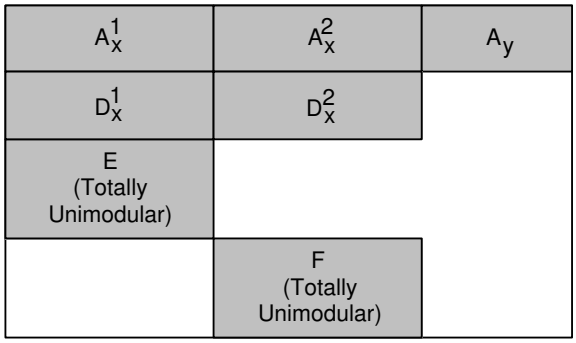

Figure 5: Each of the $|\mathcal{S}|$ ship polyhedron has a special one-ship structure.

The proof of Lemma 2.5.1 takes advantage of the well-known result of totally-unimodular (TU) matrix characteristics (e.g., Proposition III.1.2.1 in [47]). We include a partial statement that is used to show our submatrices E and F are TU.

Proposition 2.5.1 *The following statements are equivalent.*

1. *P is TU.*

2. *A matrix obtained by multiplying a row of P by -1 is TU.*

3. *A matrix obtained by interchanging two rows of P is TU.*

We also need the following famous result (e.g., Theorem 19.3(iv) in [58]) to prove Lemma 1.1.

Proposition 2.5.2 *Let P be a matrix with entries $0, +1, -1$. Then P is totally-unimodular if and only if each collection of rows of A can be split into two parts so that the sum of the rows in one part minus the sum of the rows in the other part is a vector with entries only $0, +1,$ and -1.*

Proof of Lemma 2.5.1:

The idea of proving the total-unimodularity of submatrices E and F in our one-ship problem is to split their row collections into two classes $R1$ and $R2$ so that the property in Proposition 2.5.2 is satisfied.

Submatrix E involves constraints (1), (2), (4), (5) and variables X_{isk}, Z_{ilsk}, for all $i, l \in \mathcal{I}, k \in \mathcal{K}_s$. We let each column represent a variable and observe that submatrix E has entries $0, +1, -1$.

For each $s \in \mathcal{S}$, let us first consider constraints (1) and (2) and relabel them as

$$\begin{aligned} (A_{sk}) \quad \textstyle\sum_i X_{isk} &= 1 \quad s \in \mathcal{S}, k \in \mathcal{K}_s, \\ (B_{is}) \quad \textstyle\sum_k X_{isk} &\leq 1 \quad i \in \mathcal{I} \setminus \{0\}, s \in \mathcal{S}. \end{aligned}$$

Variable X_{isk} appears once in (A_{sk}) for all $k \in \mathcal{K}_s$, and once more in (B_{is}) for all $i \in \mathcal{I}\setminus\{0\}$. Still not considering those for $i = 0$, group all rows of (A_{sk}) as $R1$ and all rows generated by constraint (B_{is}) as $R2$. Up to now, for each column in $R1 \cup R2$, we have the sum of all rows in $R1$ cancels out the sum of all rows in $R2$. For all (A_{sk}), let us now put them aside and we will consider them later.

Now, we consider constraints (4) and (5) which involve both variables X_{isk} and Z_{ilsk}. For each $s \in \mathcal{S}$, let (D_{isk}) and (E_{isk}) denote constraints (4) and (5), respectively; i.e.,

$$\begin{aligned} (D_{isk}) \qquad \textstyle\sum_l Z_{ilsk} - X_{isk} &= 0, \quad i \in \mathcal{I}, s \in \mathcal{S}, k \in \mathcal{K}_s\setminus\{K_s\} \\ (E_{isk}) \quad \textstyle\sum_i Z_{ilsk} - X_{ls(k+1)} &= 0, \quad i \in \mathcal{I}, s \in \mathcal{S}, k \in \mathcal{K}_s\setminus\{K_s\}. \end{aligned}$$

Use the following rule of assigning these constraints for each $s \in \mathcal{S}$:

- For each $i, l \in \mathcal{I}$ and $k \in \mathcal{K}_s\setminus\{K_s\}$, variable Z_{ilsk} appears once in both types of constraints. Since it is our goal to take advantage of Proposition 2.5.2, the same variable that appears in (4) has to be separated from its appearance in (5). So, for all $i \in \mathcal{I}$, assign (D_{is0}) to $R1$ and (E_{is0}) to $R2$.

- Since X_{isk} appears in both $(D_{is(k+1)})$ and (E_{isk}) with different signs for all $k \in \mathcal{K}_s\setminus\{0, K_s\}$, for these values of k we want to put $(D_{is(k+1)})$ in the same group with (E_{isk}). Then we assign constraint (D_{is1}) to $R2$ and (E_{is1}) to $R1$, for all $i \in \mathcal{I}$.

- For the same reason, we assign (D_{is2}) to $R1$ and (E_{is2}) to $R2$, for all $i \in \mathcal{I}$.

- We keep doing this cross assignment until we hit $k = K_s - 1$.

Up to this point, for each column related to variables Z_{ilsk}, we have the sum of all rows in $R1$ cancels out the sum of all rows in $R2$, for all $i \in \mathcal{I}$. And for each column related to variables X_{isk}, we have the sum of all rows in $R1$ cancels out the sum of all rows in $R2$ for all $k \in \mathcal{K}_s \backslash \{0, K_s\}$ and all $i \in \mathcal{I} \backslash \{0\}$.

For the column related to variable X_{0s0}, we have the sum of all rows in $R1$ and $R2$ equals $+1$ and 0, respectively. And for the column related to variable X_{0sK_s}, the sum of all rows in $R1$ and $R2$ are either -1 or 0. One of them will equal -1 and the other one 0.

Recall that we still have not assigned (A_{sk}), for all $k \in \mathcal{K}_s$. The variable related to these unassigned rows is X_{0sk}, for all $k \in \mathcal{K}_s$. Further,

- assign (A_{s0}) to $R2$ to have cancel out the sum of all rows in column X_{0s0} in $R1$. As a consequence of this assignment, the sum of all rows of $R1$ and $R2$ in column X_{is0} $(i \in \mathcal{I} \backslash \{0\})$ are $+1$ and 0 respectively. But this is fine as long as it does not violate the condition to apply Proposition 2.5.2.

- assign (A_{sK_s}) to whichever group has the current sum of rows in column X_{0sK_s} equal -1. This will make the row sums in column X_{0sK_s} equal 0 in both $R1$ and $R2$.

- assign the rest of (A_{sk}) with $k \in \mathcal{K} \backslash \{0, K_s\}$ arbitrarily to either $R1$ or $R2$. This will make either one of the row sums equal 1 in the corresponding columns. This is fine since we still can apply Proposition 2.5.2.

Until now, we have successfully assigned all constraints to two groups $R1$ and $R2$ so that the sum of the rows in $R1$ minus the sum of the rows in $R2$ is a vector with entries only $0, +1$, and -1. By Proposition 2.5.2, our submatrix E is totally-unimodular.

Similarly, we will now construct the respective set $R1$ and $R2$ for submatrix F which involves constraints (6), (7), (11) and variables Y_{js}, XP_{jsk}, XD_{jsk}, and XC_{jsk}, for all $j \in \mathcal{L} \cup \mathcal{U}, k \in \mathcal{K}_s$; namely,

$$
\begin{array}{lrcl}
(F_{js}) & \sum_k XP_{jsk} - Y_{js} & = & 0 \quad j \in \mathcal{U}, s \in \mathcal{S} \\
(H_{js}) & \sum_k XD_{jsk} - Y_{js} & = & 0 \quad j \in \mathcal{L} \cup \mathcal{U}, s \in \mathcal{S} \\
(KD_{jsk}) & -XC_{js(k+1)} + XC_{jsk} + XP_{jsk} - XD_{jsk} & = & 0 \quad j \in \mathcal{L} \cup \mathcal{U}, s \in \mathcal{S}, k \in \mathcal{K}_s \backslash \{K_s\}.
\end{array}
$$

For each ship $s \in \mathcal{S}$ apply the following rules: assign all constraints (F_{js}) to $R1$ for all $j \in \mathcal{U}$, assign all constraints (H_{js}) to $R2$ for all $j \in \mathcal{L} \cup \mathcal{U}$, and assign all constraints (KD_{jsk}) to $R2$ for all $j \in \mathcal{L} \cup \mathcal{U}, k \in \mathcal{K}_s \backslash \{K_s\}$.

Let us now verify that these rules do not violate the requirements of applying Proposition 2.5.2. For all unloaded cargo $j \in \mathcal{U}$, variable Y_{js} appears once in (F_{js}) and once more in (H_{js}). Hence for all columns related to variable Y_{js}, the sum of all rows in $R1$ cancels out the sum of all rows in $R2$. For all loaded cargo $j \in \mathcal{L}$, variable Y_{js} only appears in (H_{js}), but it does not violate the condition for applying Proposition 2.5.2.

Variable XP_{jsk} appears once in constraint (F_{js}) and once more for $k \in \mathcal{K}_s \backslash \{K_s\}$ in constraint (KD_{jsk}). Hence, the sum of all rows in $R1$ cancels out the sum of all rows in $R2$ for such cases. For variable XP_{jsK_s}, we have the sum equals $+1$ in $R1$ and 0 in $R2$, but it still satisfies the condition to apply Proposition 2.5.2.

Variable XD_{jsk} has a coefficient of $+1$ in constraint (H_{js}) and -1 in constraint (KD_{jsk}), for all $k \in \mathcal{K}_s \backslash \{K_s\}$. These entries result in a sum of 0 for all rows of $R2$ in each related column. For XD_{jsK_s}, the variable only appears in (H_{js}) resulting in the sum of 1 in rows of $R2$.

For variable YY_{jsk} with $k \in \mathcal{K}_s \backslash \{0, K_s\}$, it appears in (KD_{jsk}) with coefficient -1 and in $(KD_{js(k+1)})$ with coefficient $+1$, $k \in \mathcal{K}_s \backslash \{K_s\}$. Hence the sum of all rows in $R2$ is 0 for all columns related to such variables. Variable YY_{js0} appears once in (KD_{js0}) and variable YY_{jsK_s} appears once in (KD_{jsK_s}), and no condition is violated

to apply Proposition 2.5.2.

From these rules of assigning rows in submatrix F, by Proposition 2.5.2 we conclude that our submatrix F is totally-unimodular. And hence the proof is complete. ■

CHAPTER III

SOLUTION STRATEGIES AND COMPUTATIONAL EXPERIMENTS

In practical applications, our multi-ship problem takes far too long to solve for an optimal solution. For problems involving a small number of ships and cargos, it is still possible to obtain optimal solutions within a reasonable amount of time. However, a typical scheduling problems faced by real tramp shipping companies involve 8 to 20 ships and as many as 240 cargos. For this size of problem, it is almost impossible to find an optimal solution within one hour. In this chapter, we introduce three types of heuristic methods that are fast and capable of finding a feasible solution within a reasonable amount of time.

3.1 Heuristic methods to solve multi-ship problem

The first heuristic is called the Multi-Period Heuristic (MPH). The strategy for this heuristic is to divide the time horizon into several smaller periods, solve the earlier period, carry over the solution to the next period, and so on. The idea is to make the period subproblems significantly smaller than the original one. The second heuristic is called the One-Ship Heuristic (OSH). This heuristic takes as a starting point the optimal solutions of all one-ship problems, which are relatively fast to compute. Then the multi-ship schedule is constructed by fixing some cargos that appear in the solutions of more than one ship. The third heuristic we introduce is called the Set-Packing Heuristic (SPH). SPH tries to generate "good" cargo *combinations* to be served by each ship. A set packing problem is solved to choose the best combination among these cargo combinations and to construct one feasible solution to the multi-ship problem. Next, we will describe these heuristic methods in detail.

3.1.1 Multi-Period Heuristic (MPH)

We may recall that our model is expected to run over and over again whenever new information becomes available. Hence, a natural heuristic method is introduced by dividing the time horizon into some time intervals. For example, if our time horizon is five weeks and new information becomes available at the beginning of each week, then we may split the scheduling problem into 5 time intervals of one-week length. We carry the solution made in the current interval into the next interval. For the later intervals, we have more and more fixed cargo assignments and by doing this we decrease the size of our problem tremendously because we only consider certain portions of the cargos and the movements of the ships are only over certain ports.

We can further decrease the complexity of each iteration by breaking the step into several sub-iterations, each allowing only one originating port to be evaluated. For example, if a period has multiple originating ports, we may break the period into several sub-iterations by allowing only one originating port to be opened. Which originating port goes first can be determined by calculating the potential revenue associated with the port, generated by the available cargos within the port. The bigger potential revenue an originating port has, the more profitable that port is; such ports are evaluated first. Of course there are many ways in which we can implement the idea of this heuristic. One is as follows.

Let $\alpha_1, \alpha_2, \cdots, \alpha_n$ be the n points in the scheduling period that break the time horizon into $n+1$ subperiods or subintervals; namely, $[0, \alpha_1), [\alpha_1, \alpha_2), \cdots, [\alpha_{n-1}, \alpha_n)$, and $[\alpha_n, \alpha_{n+1}]$ where α_{n+1} is the time horizon. Let $\mathcal{A}_i$ denote the set of cargos j having $EPT_j \in [\alpha_{i-1}, \alpha_i)$, for all $i \in \{1, 2, \cdots, n\}$. Also, let $\mathcal{B}_s$ be the set of cargos permanently assigned to ship $s \in \mathcal{S}$ and let $\mathcal{R}$ be the set of routes for all ships. Note that the size of $\mathcal{R}$ is the same as the number of ships, meaning that each ship has one corresponding route in $\mathcal{R}$. Our multi-period heuristic (MPH) algorithm can be described as follows.

Algorithm 3.1.1

Initialization:

$\mathcal{B}_s = \mathcal{L}_s$, $\mathcal{R} = \emptyset$;

begin algorithm

for (each $i = 1$ **to** n) **do**

Solve multi-ship model with $\mathcal{U} = \mathcal{A}_i, \mathcal{L}_s = \mathcal{B}_s$;

$\mathcal{B}_s \leftarrow \mathcal{B}_s \cup \{j : j \in \mathcal{A}_i$ that has been assigned to ship s by the iteration$\}$;

Update $\mathcal{R}$;

end for

Solve multi-ship model one more time, with $\mathcal{U} \leftarrow \mathcal{U} \backslash \bigcup_{s \in \mathcal{S}} \mathcal{B}_s$ and $\overline{\mathcal{L}}_s = \mathcal{B}_s$, and $\mathcal{R}$;

end algorithm

3.1.1.1 Example

We solve our 10-ship example problem using this proposed heuristic method. Table 7 shows the step-by-step process until we get the initial schedule for each ship, with some permanent assignment of the served cargos. Using this initial schedule, we run the problem one more time to allow more unloaded cargos to be picked up. This last step in fact gives us a more profitable schedule as shown in Table 8 and we have determined the final schedule for our multi-ship problem. The final schedule is depicted in Figure 6 and the total computational time to complete the iterations in MPH is 764.06 seconds. It takes an additional 126.22 seconds to run the model one more time to allow more cargos to be served, yielding a total time of 890.28 seconds to run the complete MPH heuristic for our particular 10-ship example.

Table 7: Step-by-step multi period heuristic applied to 10-ship problem example

#Itr	Period	Ports opened (Origin)	Ports opened (Destination)	Cargos Available	Assignment
1	14-17 Apr	27	5	20	-
2	18-20 Apr	27	10	26	-
				27	-
				28	-
				29	-
3	21 Apr	21	22	36	Ship-10
		27	25	17	Ship-4
				18	Ship-4
				19	Ship-4
		31	15, 24	37	-
				38	-
4	22 Apr	14	24, 31	21	Ship-4
				22	Ship-4
				23	Ship-4
				24	Ship-4
				25	Ship-4
5	23 Apr	12	20, 23	9	-
				10	Ship-4
6	24 Apr	6	11, 24, 28, 36	39	-
				40	Ship-5
				41	Ship-5
				42	-
7	25 Apr	12	1, 3, 8, 11, 20, 24-26, 28	1	Ship-9
				2	Ship-9
				3	-
				4	Ship-9
				5	-
				6	-
				7	-
				8	-
		27	13, 17	30	-
				31	-
				32	-
				33	-
8	26-30 Apr	31	18, 20, 24	11	Ship-8
				12	Ship-8
				13	Ship-8
				14	Ship-8
				15	Ship-8
				16	Ship-8
9	1-5 May	14	3	34	-
				35	-

Table 8: Final MPH cargo assignment to 10-ship problem

Ship	Additional cargo	Cargos served	Total profit (USD)
1	-	43-52	52,289
2	-	53-55	19,381
3	-	56	62,908
4	-	10, 17-19, 21-25, 57	149,439
5	-	40-41, 58-61	174,627
6	-	62	73,311
7	-	63-67	108,809
8	38	11-16, 38	54,037
9	-	1-2, 4, 68-74	135,182
10	-	36, 75-79	216,801
		Total unloaded cargos served: 21	1,046,784

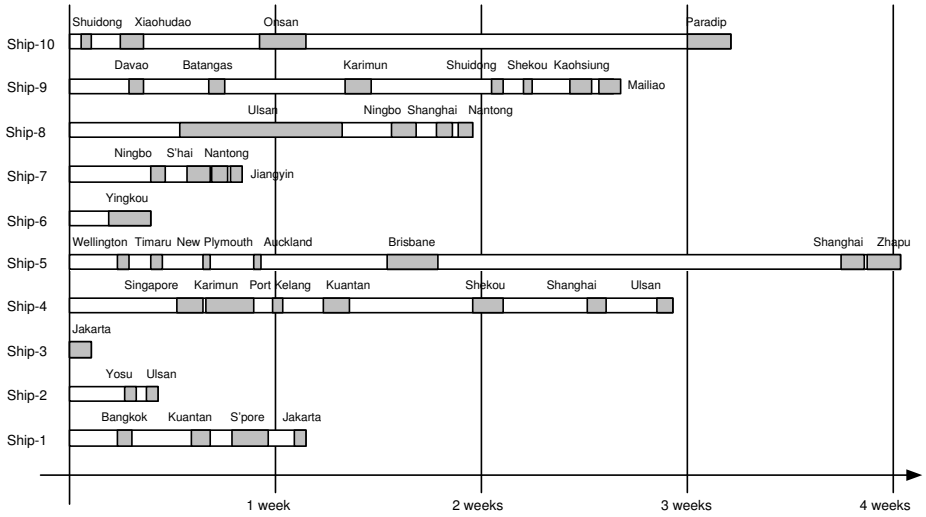

Figure 6: Schedule generated by multiperiod heuristic

3.1.2 One-Ship Heuristic (OSH)

Here we use the optimal one-ship solutions as a starting point to OSH. First, we run the problem for each ship separately. Once we have found all one-ship optimal solutions, we may discover that some cargos appear on only one ship, while others appear on multiple ships. The latter case is clearly infeasible for the multi-ship problem.

These one-ship models allow complete freedom to the ships to serve any unloaded cargo and at the same time, allow multiple ships to serve a given cargo. If a cargo appears on only one ship, it makes good sense to assign it permanently to that ship, since we may somewhat reduce the complexity of the problem. We call these kind of cargos as 1-ship cargos. When multiple ships are trying to serve the same cargo, we must identify the best ship to serve that particular cargo. We name such cargo as multi-ship cargo.

We may come up with several methods in finding the best ship to assign multi-ship cargos. In their original paper, Jetlund and Karimi [32] proposed a method of assigning a cargo to the ship on which it has the maximum marginal profit. The marginal profit is defined as the profit that the ship loses for not serving the cargo. So, the marginal profit gives a measure of how important one particular cargo is to a serving ship. They also introduce another heuristic that progressively assigns all cargos permanently to ships by using the number of bidding ships as a guide. They fixed all 1-ship cargos first, then 2-ship cargos, and so on.

We propose another scheme similar to the second heuristic proposed in [32]. Again, for 1-ship cargos, it makes perfect sense to assign them to their corresponding ships. Then we make a list of multi-ship cargos in decreasing order of number of bidding ships. The more bidding ships we have for one particular cargo, the more favorable that cargo becomes. Then it makes sense to select the best ship for the most favorite cargo first, followed by the next less favorable cargo, etc. We resolve only a few cargo bid conflicts at a time by solving the multi-ship problem with the unloaded

cargos limited to the ones being decided. In each iteration we have some cargos being assigned to some ships. In each iteration, the ship routes are reoptimized to accommodate the newly assigned cargos. Thus, we gradually assign all cargos permanently to ships. The last step is to run the multi-ship model one more time with all the multi-ship cargos assigned permanently. Here we try to see if more cargos, that do not even appear in the optimal one-ship solutions, can be served.

After we solve the one-ship problem optimally for all ships, we observe the optimal results and categorize the cargos as 1-ship, 2-ship, ..., n-ship, according to the number of competing ships. Let $\alpha \in \{0, 1, \cdots, n\}$ indicate the number of competing ships. Let $\mathcal{A}_\alpha$ be the set of α-ship cargos, $\alpha \in \{0, 1, \cdots, n\}$. Let $\overline{\mathcal{U}}$ and $\overline{\mathcal{L}}_s$ be the set of unloaded and loaded cargos that have to be served in solving the multi-ship problem; i.e., $\sum_{s \in \mathcal{S}} Y_{js} = 1$, for all $j \in \overline{\mathcal{U}}$ and $\overline{\mathcal{L}}_s$ respectively. Also, let $\mathcal{B}_s$ denote the set of cargos that have been permanently assigned to a ship s. Clearly, the set $\mathcal{B}$ grows as we progress. To initialize, we let the set of loaded cargo $\mathcal{L}_s$ be the initial $\mathcal{B}_s$ for every s. Lastly, we let $\mathcal{R}$ denote the current routes for all $s \in \mathcal{S}$. Note that $\mathcal{R}$ has as many members as the number of ships in the problem. Using this notation, our one-ship heuristic (OSH) algorithm is as follows.

Algorithm 3.1.2

Initialization:
$\mathcal{B}_s = \mathcal{L}_s$, $\mathcal{R} = \emptyset$;

begin algorithm

Solve one-ship problem for all ships $s \in \mathcal{S}$

Categorize cargos as α-ship cargo;

Construct sets $\mathcal{A}_\alpha$ and place cargos in respective sets;

$\alpha_{max} = \max\{\alpha : \mathcal{A}_\alpha \neq \emptyset\}$;

for (each $j \in \mathcal{A}_1$) **do**

Let $\overline{s}$ be the ship to which cargo j is assigned in the one-ship optimal solution;

$\mathcal{B}_{\overline{s}} \leftarrow \mathcal{B}_{\overline{s}} \cup \{j\}$;

end for

for ($\alpha = \alpha_{max}$ **to** 2) **do**

if ($\mathcal{A}_\alpha \neq \emptyset$ **then**

for (each $j \in \mathcal{A}_\alpha$ **do**

Solve multi-ship model with $\overline{\mathcal{U}} = j$, $\overline{\mathcal{L}}_s = \mathcal{B}_s$;

Let $\overline{s}$ be the ship to which cargo j;

$\mathcal{B}_{\overline{s}} \leftarrow \mathcal{B}_{\overline{s}} \cup \{j\}$;

Update $\mathcal{R}$;

end for

end if

end for

Solve multi-ship model one more time, with $\mathcal{U} \leftarrow \mathcal{U} \backslash \bigcup_{s \in \mathcal{S}} \mathcal{B}_s$, $\overline{\mathcal{L}}_s = \mathcal{B}_s$, and $\mathcal{R}$;

end algorithm

3.1.2.1 Example

Table 9 shows a step-by-step construction of a multi-ship schedule for our problem. With all multi-ship cargo assigned to their respective ships, we run the program one more time to see if any ship would serve any more unloaded cargos. And there are improvements since more cargos are served, as shown in Table 10. The final schedule generated by this heuristic is depicted in Figure 7. The total time to run OSH for the 10-ship problem, excluding the time to obtain 1-ship optimal solutions, is 134.54 seconds. Recall that the ten 1-ship problems were solved to optimality within a time range between 5.18 seconds and 240.53 seconds. The total time to run OSH, including the time to solve 1-ship problems, is $1,364$ seconds.

Table 9: Step-by-step one ship heuristic applied to 10-ship problem example

# Iteration	Cargo	Types of cargo	Bidding ships	Ship Assigned
1	6	1-ship	3	3
	22	1-ship	1	1
	30	1-ship	3	3
	31	1-ship	3	3
	32	1-ship	3	3
	40	1-ship	5	5
	41	1-ship	5	5
2	13	5-ship	2, 6, 7, 8, 9	8
	14	5-ship	2, 6, 7, 8, 9	8
3	11	4-ship	2, 6, 8, 9	8
	12	4-ship	2, 6, 8, 9	8
	38	4-ship	2, 6, 7, 8	8
4	2	3-ship	1, 3, 4	1
	15	3-ship	2, 6, 8	8
	17	3-ship	1, 3, 4	1
5	1	2-ship	3, 4	4
	5	2-ship	1, 4	1
	9	2-ship	1, 4	4
	16	2-ship	6, 8	8
	18	2-ship	1, 3	1
	19	2-ship	1, 3	1
	21	2-ship	1, 4	1
	25	2-ship	1, 4	1
	36	2-ship	2, 10	10

Table 10: Final OSH cargo assignment to 10-ship problem

Ship	Additional cargos	Cargos served	Total profit (USD)
1	23-24	2, 5, 17-19, 21-25, 43-52	135,971
2	-	53-55	19,381
3	8, 10, 34-35	6, 8, 10, 30-32, 34-35, 56	78,432
4	3-4, 7	1, 3-4, 7, 9, 57	94,569
5	-	40-41, 58-61	174,627
6	-	62	73,311
7	-	63-67	108,809
8	-	11-16, 38	54,037
9	-	68-74	129,188
10	-	36, 75-79	216,801
		Total unloaded cargo served: 32	1,085,126

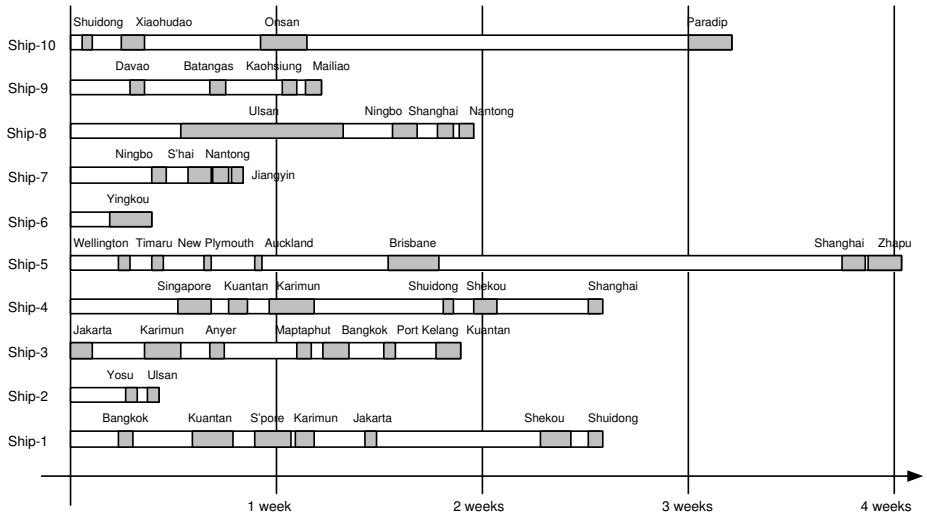

Figure 7: Schedule generated by one ship heuristic

3.1.3 Set-Packing Heuristic (SPH)

The idea of this heuristic, as any set packing method, is to construct some, if not all, possible routes that each ship could follow and then choose one for every ship such that the profit is maximized. Obviously, if we can generate *all* possible routes that each ship could follow, we will definitely get the optimal solution. However, generating all possible routes might be too costly. So, another heuristic we propose is to make a list of "good" cargo combinations to be served by a ship and let our model decide if each combination is feasible for the ship. If the combination is feasible, our multi-ship model also determines what the optimal route is for that particular combination. Hence, a ship's route is an output given by the multi-ship model. Certain cargo combinations might not even be feasible to the problem because the time window requirements could not be met, or the capacity limit is violated, or any one of many other possible reasons. We only populate all cargo combinations that are feasible to our problem and choose the best cargo combination for each ship by formulating a set packing problem.

Let N_s be the maximum number of possible cargo combination for ship s and let M_s be the maximum number of cargo in one particular combination for ship s. Also,

let $\mathcal{R}_s$ be the collection of routes for ship s. Clearly, if we set N_s to be infinity and M_s to equal as many as available cargos, then we may have *all* possible sets of cargo combinations. It means that for every ship s, the set $\mathcal{R}_s$ would contain all possible cargo combinations. Therefore, the set packing problem will find the optimal solution to the problem.

By controlling N_s and M_s we determine how big the set $\mathcal{R}_s$ is for every ship s. The bigger $\mathcal{R}_s$ we want, the longer computational time we need to spend to construct possible cargo combinations. Suppose we are given the sets $\mathcal{R}_s$ for all ship s, and for every cargo combination $r \in \mathcal{R}_s$, we have c_{rs} as the revenue generated by ship s by choosing combination r. Let x_{rs} be an indicator variable which equals 1 if combination $r \in \mathcal{R}_s$ is served by ship s and 0 otherwise. The set packing problem can be written as follow:

$$
\begin{array}{lrrl}
(SP) \quad \text{Max} & \sum_{s\in\mathcal{S}} \sum_{r\in\mathcal{R}_s} c_{rs}\, x_{rs} & & \\
\text{s.t.} & \sum_{s\in\mathcal{S}} \sum_{r\in\mathcal{R}_s} a_{jrs}\, x_{rs} & \leq & 1, \quad \text{for all } j \in \mathcal{U} \cup \mathcal{L} \\
& \sum_{r\in\mathcal{R}_s} x_{rs} & = & 1, \quad \text{for all } s \in \mathcal{S} \\
& x_{rs} \in \{0,1\} & &
\end{array}
$$

where a_{jrs} is an indicator that equals 1 if cargo j is in combination r of ship s, and 0 otherwise.

3.1.3.1 Generating feasible cargo combinations

In this subsection we will particulary look at how we generate the cargo combinations for the set packing heuristic. Suppose we have a set of loaded and unloaded cargos, $\mathcal{L}$ and $\mathcal{U}$ respectively, and $\mathcal{L} = \bigcup_{s\in\mathcal{S}} \mathcal{L}_s$. At time zero, each ship s has its own loaded cargos on-board, $\mathcal{L}_s$. If we solve the multi-ship model ignoring all unloaded cargos, then we will have a route that will deliver all on-board cargos. Hence the first cargo combination is constructed using only on-board cargos at time zero. We may then add one unloaded cargo $j \in \mathcal{U}$ to the set and re-solve our model by allowing the ship to

only serve cargos in $\mathcal{L}_s \cup \{j\}$. If the problem is feasible, it will find another cargo combination to our collection. Otherwise, we may ignore this combination and proceed by adding another cargo $j' \in \mathcal{U} \backslash \{j\}$ and re-solve our model. The infeasibility could happen if certain cargo combinations violate some constraints; e.g., time-windows are not met, ship is overloaded, or simply that there is not enough journey legs to serve such combinations. For feasible cargo combinations, we may then add more cargo, one at a time, to come up with new combinations, as long as we do not violate the maximum number of cargo M_s and the maximum number of combinations N_s that we defined earlier. These steps of generating possible combinations for a ship $s \in \mathcal{S}$ are summarized as follow.

Algorithm 3.1.3

Initialization:
$\mathcal{A} = \emptyset, \mathcal{B} = \emptyset, \mathcal{C} = \mathcal{L}_s, \mathcal{D} = \mathcal{U}$;
$n = 0$;

begin algorithm

Solve one-ship problem for ship s with $\mathcal{U} = \emptyset$;
$\mathcal{B} \leftarrow \mathcal{B} \cup \{\mathcal{C}\}$;
while ($\mathcal{B}$ is not empty) **do**
Take $\mathcal{C}$, the next member of $\mathcal{B}$;
$\mathcal{B} \leftarrow \mathcal{B} \backslash \{\mathcal{C}\}$;
$\mathcal{A} \leftarrow \mathcal{A} \cup \{\mathcal{C}\}$;
$n \leftarrow n + 1$;
for (each $j \in \mathcal{D} \backslash \mathcal{C}$) **do**
$\mathcal{C} = \mathcal{C} \cup \{j\}$;
$m = |\mathcal{C} \backslash \mathcal{L}_s|$;
if ($m < M_s$ and $n < N_s$) **then**

Solve one-ship problem for ship s with $\mathcal{U} = \mathcal{C} \backslash \mathcal{L}_s$;

if (the problem is solvable) **then**

$\mathcal{B} \leftarrow \mathcal{B} \cup \{\mathcal{C}\}$;

end if

end if

end for

end while

return $\mathcal{A}$, the set of feasible cargo combinations;

end algorithm

Clearly, this algorithm is eventually finite, since the number of possible cargo combinations is finite. Each time we solve a one-ship model for a specific cargo combination, we also get the optimal route associated with that cargo combination, which we may store for further needs. Furthermore, the unloaded cargos can be sorted in a non-increasing order of potential revenue. Hence, we can most likely generate cargo combinations containing cargos with high profitability in the beginning of our search.

3.1.3.2 Generating priority list for SPH

The way we generate a priority list is crucial for the SPH method to work well. A different priority list might lead to different solutions once we put a limit on the number of cargo combinations to be generated, or on the number of cargos within each combination. Here we discuss a way to generate the priority list based on the revenue and cost analysis for each cargo. First of all, we group cargos based on their pickup-destination ports and availability. It is likely for a ship to pickup cargos whose availabilities are similar and whose pickup and destination ports are the same. Hence it is natural to group them together and to analyze them altogether when we evaluate a cargo combination. So instead of adding cargos belonging to the same group one by

Table 11: Final SPH cargo assignment to 10-ship problem

Ship	Cargos served	Total profit (USD)
1	2, 5, 9, 17-19, 21-22, 25, 43-52	162,117
2	53-55	19,381
3	56	62,908
4	6, 10, 30-32, 57	128,014
5	40-41, 58-61	174,627
6	62	73,311
7	63-67	108,809
8	11-16, 38	54,037
9	68-74	129,188
10	36, 75-79	216,801
	Total unloaded cargos served: 23	1,129,193

one, we add them all at one time. Here we may decrease the number of possible cargo combinations. For example, in our 10-ship problem, we have a total of 30 groups of unloaded cargos as opposed to 42 single unloaded cargos.

After we group the cargos as we explain in the foregoing discussion, we create a priority list of cargo groups based on their profitability. The profitability is approximated as the revenue of serving a cargo group minus the fuel and time-charter costs to travel from the pickup to the download port.

3.1.3.3 Example

We apply the foregoing idea to solve our 10-ship example problem by SPH. We limit the maximum number of feasible cargo combinations to 1000 and the maximum number of cargo groups within each combination to 5. The SPH solution to our 10-ship problem is \$1,129,193. Table 11 shows the cargo-to-ship assignment generated by the set packing heuristic. The computational time is $3,146$ seconds and the final schedule is depicted in Figure 8.

Optimal solution of the 10-ship problem example

If we treat each cargo as a single cargo, as opposed to creating some cargo groups, and allow the SPH to perform exhaustive examination of every possible cargo combination,

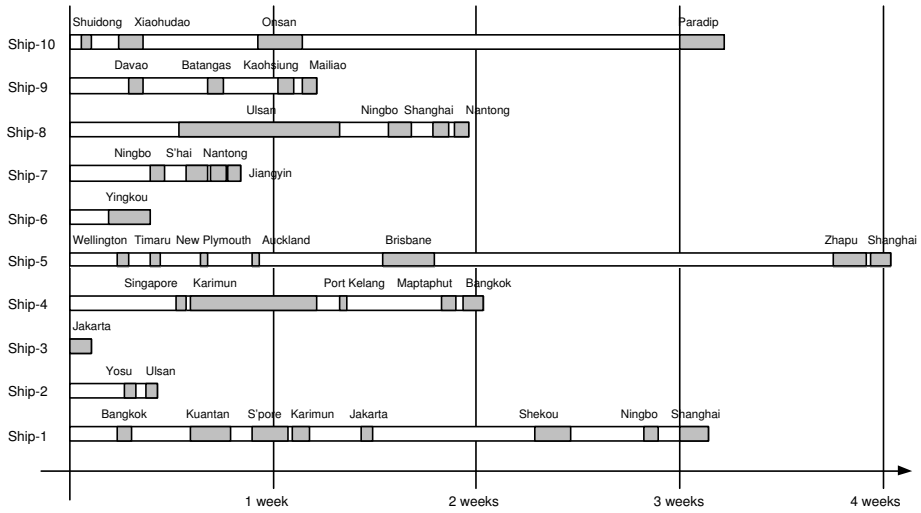

Figure 8: Schedule generated by set-packing heuristic

Table 12: Number of possible cargo combinations for each ship

Ship	Total legs	Total must-visit ports	First destination	# Feasible cargo combinations
1	8	4	Bangkok	25,484
2	8	2	Yosu	21,649
3	7	1	Tj. Priok	127,062
4	7	2	Singapore	300,649
5	8	4	Wellington	17
6	8	1	Yingkou	6,718
7	7	4	Ningbo	272
8	7	1	Ulsan	1,040
9	7	4	Davao	272
10	7	2	Shuidong	16,774

we will eventually find the optimal solution to our 10-ship problem. The number of possible cargo combinations for each ship in our 10-ship problem is given in Table 12. After solving the set-packing problem, which is found by CPLEX at the root node in 0.34 second, we obtained the optimal solution for our 10-ship problem, \$1,137,154. The computational time is $62,067$ seconds. The optimal cargo-to-ship assignment is given in Table 13 and the final schedule is shown in Figure 9.

Table 13: Optimal cargo assignment to 10-ship problem

Ship	Cargos served	Total profit (USD)
1	2, 5, 9, 17-19, 21-22, 25, 43-52	162,117
2	53-55	19,381
3	56	62,908
4	6, 8, 10, 30-32, 34-35, 57	135,975
5	40-41, 58-61	174,627
6	62	73,311
7	63-67	108,809
8	11-16, 38	54,037
9	68-74	129,188
10	36, 75-79	216,801
	Total Unloaded Cargos served: 26	1,137,154

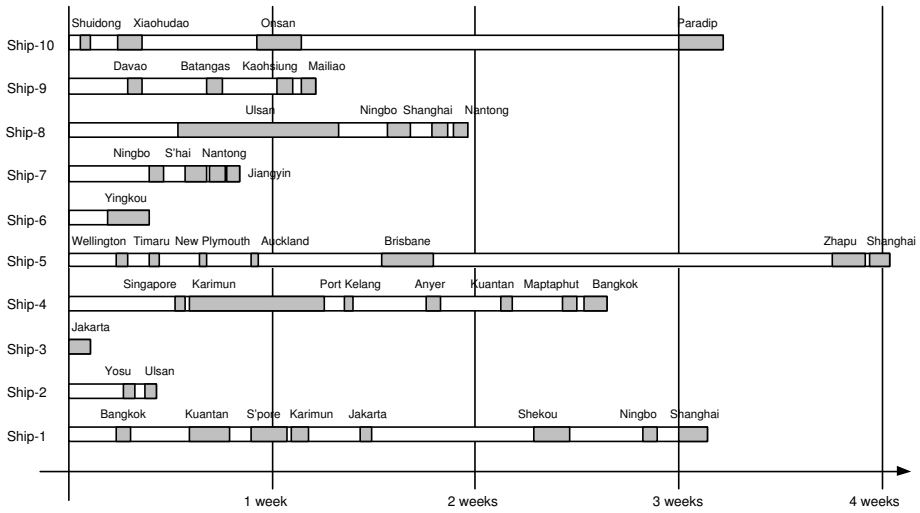

Figure 9: Optimal schedule for the 10-ship problem example

3.2 Pickup and delivery problem with time-windows

As previously mentioned, our problem can be formulated as a pickup and delivery problem with time-windows constraints. One of the solution strategies is to formulate the problem as a set packing problem. Here we try to implement full column generation where each column represents a possible route for a ship to follow. But let us first explain how the network is constructed.

3.2.1 Network construction

At time zero, each ship s is located at a certain place, which could be a port or in the middle of the sea heading towards a predetermined port. Without loss of generality let us assume that each ship s is at a port which we refer to as the origin port. We have several on-board cargos $\mathcal{L}_s$ on ship s at time zero and there are also unloaded cargos $\mathcal{U}$ distributed among ports, each with its origin and destination information.

Let there be $n = |\mathcal{U}|$ unloaded cargos indexed by j. Associate a node j to the pickup location of cargo $j \in \mathcal{U}$ and to its delivery location associate a node $(n + j)$. For each ship s, we also have a list of $m = |\mathcal{L}_s|$ destinations of on-board cargos. Associate a node $(2n + j)$ to the download location of cargo $j \in \mathcal{L}_s$. Furthermore, we also have nodes 0 and $(2n + m + 1)$ associated with the origin and the dummy ports, respectively.

Therefore, for each ship s we have a set of nodes N in the network where $N = \{0, 1, 2, ..., n, n + 1, n + 2, ..., 2n, 2n + 1, 2n + 2, ..., 2n + m, 2n + m + 1\}$. Let $P^+ = \{1, 2, ..., n\}$, $P^- = \{n + 1, n + 2, ..., 2n\}$, and $Q = \{2n + 1, 2n + 2, ..., 2n + m\}$ so that $P^+ \cup P^- \cup Q$ is the set of nodes other than the origin and the dummy ports.

Our computational network consists of nodes in the set $N = P^+ \cup P^- \cup Q \cup \{origin, dummyport\}$ and sets of feasible arcs. Note that many nodes in our network might refer to the same port in the physical network. There are arcs among nodes in each group P^+, P^-, and Q. There are also 11 groups of arcs between two different groups depicted in Figure 10 connecting two different node-groups. All of these arcs

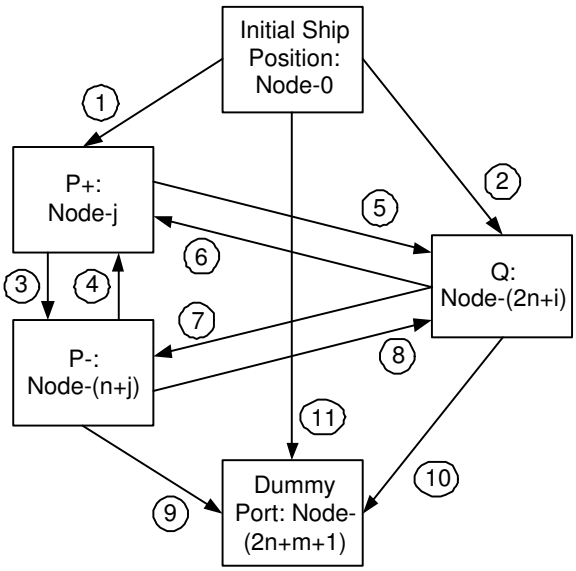

Figure 10: Network for PDPTW approach

are mostly defined by movements between nodes and the time-window requirement for each node.

Arc-group number-1 connects the initial ship position, node 0, to some nodes in P^+. The arc is defined from node 0 to node $j \in P^+$ if a ship traveling to node j meets the time window *and* if the ship continues to travel to node $(2n + i) \in Q$ right after it visits node j, the download time window in node i is still met *for all* i. Arc-group number 2 is defined from node 0 to *all* nodes in Q.

Some members of arc-group number 3 are connecting every pickup port in set P^+ to the corresponding download port in set P^-. But there are also arcs connecting a pickup port of cargo j_1 in P^+ to a download port of cargo j_2 in P^-. This happens if the ship serving both cargos picks-up j_2 and then picks-up j_1 while cargo j_2 is still on board. Hence, this kind of arc is defined if the arc connecting the pickup port of j_2 and the pickup port of j_1 in P^+ is defined *and* if the download time window of cargo j_2 can still be met after the ship travels from the pickup port of cargo j_1 in P^+ to the download port of cargo j_2 in P^-.

Arc-group number 4 is defined from a download port of cargo j_1 in P^- to a

pickup port of cargo j_2 in P^+ if after the ship under consideration downloads cargo j_1, it travels to the pickup port of cargo j_2 and the pickup time window can still be satisfied. Arc-group number 5 is from a pickup port of cargo $j_1 \in \mathcal{U}$ in P^+ to a download port of cargo $j_2 \in \mathcal{L}$ in Q. An arc in this group is defined if after the ship under consideration picks-up cargo j_1 and continues its voyage to download cargo j_2, provided the download time-window for j_2 is still met.

The next arc-group number 6 is from a download port of cargo $j_1 \in \mathcal{L}$ in Q to a pickup port of cargo $j_2 \in \mathcal{U}$ in P^+. An arc in this group is defined if after the ship under consideration downloads cargo j_1 and continues its journey to pickup cargo j_2, provided the pickup time-window can still be met.

Arc-group number 7 is from nodes in Q to nodes in P^-. An arc from a download node of cargo $j_1 \in \mathcal{L}$ in Q to a download node of cargo $j_2 \in \mathcal{U}$ in P^- is defined when cargo j_2 is onboard when the corresponding ship visits the download node of cargo j_1 or when the corresponding arc from the pickup port of cargo j_2 in P^+ to the download port of j_1 in Q is defined. In other words, an arc in group number 7 is defined when the corresponding arc in group number 5 is also defined. The reverse arc-group number 8 from nodes in P^- to nodes in Q is defined whenever the download time-window in Q is still met after the ship travels from P^-.

Arc-groups $9, 10$, and 11 indicate the end of the voyage for a ship. Arc-groups 9 and 10 indicate that the ship may be left idle after it visits a download port of some loaded cargos in Q or a download port of some unloaded cargos in P^-, respectively. Arc-group number11 indicates that the ship does not have onboard cargos at the beginning of the time horizon *and* is not used at all during the time horizon.

We also have another type of arc connecting two nodes within a group. Let us take a look at each group and explain the arcs within each of them. An arc between two nodes in Q is defined from the tail-node to the head-node if the download time-window in the head-node can still be met after the ship downloads some cargos in the tail-node and travels from the tail-node to the head-node.

To see how an arc between two pickup nodes in P^+ is defined, let us suppose that P_1 and P_2 are the pickup nodes of cargos j_1 and j_2, respectively, where $P_1, P_2 \in P^+$ and also suppose that $D_1, D_2 \in P^-$ are the corresponding download nodes. Then an arc between two pickup nodes in P^+ from P_1 to P_2 is defined if $P_1 \longrightarrow P_2 \longrightarrow D_1 \longrightarrow D_2$ or $P_1 \longrightarrow P_2 \longrightarrow D_2 \longrightarrow D_1$ is a feasible route for the ship under consideration. Similarly, an arc between two pickup nodes in P^- from D_1 to D_2 is defined if $P_1 \longrightarrow P_2 \longrightarrow D_1 \longrightarrow D_2$ or $P_2 \longrightarrow P_1 \longrightarrow D_1 \longrightarrow D_2$ is a feasible route for the ship under consideration.

After we have sets of nodes and arcs for a ship, a possible path from the origin (node 0) to any nodes in P^-, Q, or the dummy node yields a possible route for the ship.

3.2.1.1 Limiting the number of possible routes

Obviously, a full column generation approach deals with a large number of possible routes. To reduce the number of possible routes, we impose some restrictions, that are:

(i) limit the maximum waiting time in a port,

(ii) limit the maximum travel time from one location to the next destination,

(iii) limit the maximum number of physical ports visited (which is also the maximum number of legs allowed in the problem),

(iv) avoid unprofitable moves by requiring that the value of potential revenue minus potential cost to be above a certain threshold profit,

(v) avoid multiple visits to the same port in one planning horizon,

(vi) limit the maximum number of new cargos served.

From the initial location of the ship, we perform a Depth First Search (DFS) and store all possible routes information along with their profits. Whenever we reach a

node that violates at least one of the restriction above, we prune that node. We also prune the nodes that violate time-windows requirements.

One problem arises when we have to deal with *symmetric* cargos. Symmetric cargos are cargos having the same pickup origins and discharge locations, even though they may have different weights or volumes. These cargos cause problems since we represent each cargo pickup location as a unique node. The same is true for cargo download locations. Hence, the sequence of visiting pickup ports (or download ports) of symmetric cargos really matters in column generation while it does not in the final physical route. This may generate multiple columns referring to the same physical route, and increase the search effort tremendously. We try to limit the effect of symmetric cargos by eliminating duplicate efforts in DFS. For example, suppose nodes A and B refer to the same pickup ports of symmetric cargos. When the DFS process reaches node A (before visiting B) and continues to B, we keep that move in our record so that later when the DFS reaches port B (before visiting A), the DFS does not continue to node A again since it is a duplicate effort.

During the DFS process, we do not store the new column containing the same cargo set as the last one found since they are corresponding to the same column. However, we only update the objective if the newly found column has a better profit than the previous one. This reduces the number of columns we store tremendously.

3.2.2 Full column generation for 10-ship example

We apply the foregoing ideas to generate columns for each ship in the 10-ship problem example. The results are shown in Table 14.

We compare the results we obtained by full column generation for each ship with the optimal solution of the 1-ship problems we showed in Table 5. For all ships, the columns generated contain the optimal 1-ship solution.

Optimal solution of the 10-ship problem example

Table 14: Columns generated for each ship

Ship	#legs	# must-visit ports	# other ports can be visited	# Routes generated	Time (secs)	Maximum profit	% of optimal
1	8	4	4	10,256	1,990.45	158,412	100%
2	8	2	6	8,906	1,530.37	191,288	100%
3	7	1	6	21,135	583.00	144,742	100%
4	7	2	5	11,171	536.74	195,108	100%
5	8	4	4	40	3.21	174,627	100%
6	8	1	7	1,217	318.17	138,912	100%
7	7	4	3	608	16.89	115,221	100%
8	7	1	6	264	9.02	46,841	100%
9	7	4	3	692	347.08	160,653	100%
10	7	2	5	156	3.00	215,735	100%

Given all possible columns shown in Table 14, we run the set packing problem to find the optimal solution to the 10-ship problem. We let CPLEX solve it under its default options. We find the optimal solution at node-0 and it found the optimal solution in 0.54 seconds. The optimal solution match the one found by set packing method described in Section 3.1.3.3. Full enumeration runs very fast (total $5,338$ seconds) compared to the time spent to run full cargo combination (total $62,067$ seconds) as discussed in Section 3.1.3.3. The time difference, of an order of magnitude, is caused by the expensive efforts in generating all possible cargo combinations.

3.3 Computational experiments for heuristic methods

In this section, we compare the performance of the three heuristics proposed in an earlier section. We create randomly generated test problems and compare the performance of the heuristics on these problems.

3.3.1 Creating random problems

The problems we create are not completely random since we take as-is port information and do not add new nor eliminate old ports. So we randomly generate the problems taking place at predefined sets of ports. The costs each ship has to pay when it visits ports are also fixed depending on the size of the ship.

The number of ships lies between the predefined minimum and maximum numbers. In our computational instances, we evaluate the problems on 3 to 18 ships. Each ship has a random maximum capacity between $5,000$ to $11,000$ dead-weight tonnages and a random time-charter cost. The bigger size the ship has, the more expensive the time-charter cost for that ship is. Moreover, each ship has a random first destination which is reached within a random number in the interval of $[0, 3.5]$ days. The determination of the number of legs for every ship is a little tricky. While we want the problem to be as random as possible, we also want the problem to be feasible. So, the number of legs for each ship is chosen to be the maximum of the total number of destinations of the loaded cargos on the ship and the number of legs generated randomly. The selection will insure that the problem will indeed be feasible.

The number of cargos is random depending on the number of ships under consideration. The more ships involved, the more cargos generated. Generally, the number of cargos are 6 to 12 times the number of ships. Cargos are created one by one. Once a cargo is created, it is randomly classified as either loaded or unloaded cargo. If it is classified as a loaded cargo, another random number is used to determine which ship it is loaded on. If the ship capacity permits the new cargo to be assigned as a loaded cargo to the ship, then we do so. Otherwise, we treat the cargo as an unloaded one. For every unloaded cargo, a port where the cargo originates is randomly selected. Each cargo also has its own random shipment-size and download port. However, the shipment rate is determined by which port the cargo originates from and how long the journey takes from its origin to its destination. Pickup time-windows are also determined randomly within the horizon and has a random length of 4 to 7 days. The earliest download time-windows for each cargo is set to be as soon as possible after the cargo is picked-up and once a cargo is picked-up, it has to be delivered in no more than 6-week period.

3.3.2 Computational results

We create random problems for $3, 4, ..., 18$ ships. For problems with 3 to 15 ships, we manage to create 80 problems each. Due to computationally expensive efforts in getting the optimal solutions for problem with a larger number of ships, we only manage to solve 30 random 14-ship problems, and 10 random problems each for $15, 16, 17$, and 18-ship problems. The problems are solved by CPLEX 9.1 using one processor on a four-CPU Sun E450 server machine under all of the default options of CPLEX. Besides the heuristics, we also run CPLEX to obtain the optimal solution. We record the profits generated by MPH, OSH, SPH, and the optimal solutions, along with their computational time.

Table 15 and Figure 11 show the quality of the heuristic solutions generated by Multi-Period Heuristic (MPH), One-Ship Heuristic (OSH), and Set-Packing Heuristic (SPH). The percentage shows the performance compared to the optimal profit and is calculated as:

$$\frac{\text{Heuristic profit}}{\text{Optimal profit}} \times 100\%.$$

Table 15: Heuristic Solution Quality

Total Ships	Total Problems	Average MPH Quality	Average OSH Quality	Average SPH Quality
3	80	93.85%	94.82%	99.20%
4	80	89.05%	96.12%	96.85%
5	80	93.66%	91.96%	95.30%
6	80	81.42%	94.78%	94.25%
7	80	87.86%	93.90%	92.26%
8	80	85.63%	91.97%	90.17%
9	80	86.63%	92.65%	91.81%
10	80	90.79%	94.95%	93.76%
11	80	80.74%	92.97%	90.46%
12	80	85.55%	92.79%	91.21%
13	80	91.56%	93.62%	95.55%
14	30	85.51%	93.42%	92.08%
15	10	83.37%	91.46%	90.30%
16	10	84.83%	93.80%	92.35%
17	10	82.74%	87.64%	89.27%
18	10	88.85%	94.66%	95.12%

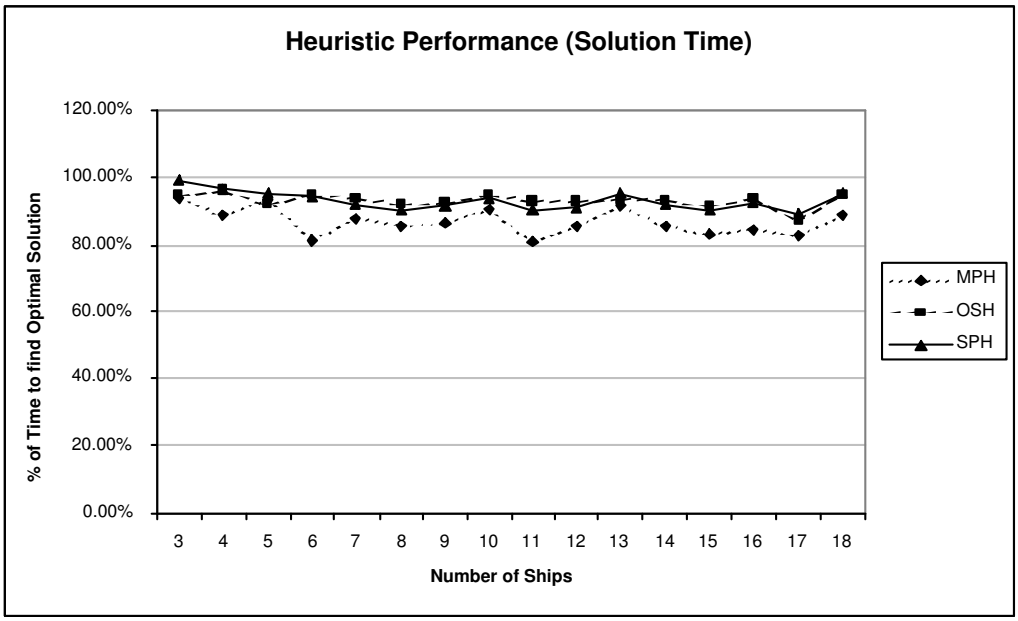

Figure 11: Graph for Heuristic Solution Quality

Based on the performance test results, we observe that all proposed heuristics reach profits above 80% of the optimal solutions in all cases. MPH performs slightly worse than the other two. In almost all cases, OSH and SPH create solutions above 90% of the optimal. We limit the number of cargo combinations created in SPH to be at most 300 combinations consisting of at most 4 new cargos, whichever is reached first. Clearly, by increasing the number of possible cargo combinations created in SPH, we may obtain a better SPH performance. All heuristics can maintain their performance as the problem becomes harder (as the numbers of ships and cargos increase). It is interesting to see that the performance of OSH is good and stable enough. In many cases, OSH slightly outperforms SPH, not only in the solution quality but also in the computational time.

Table 16 and Figure 12 show the average time spent in finding the heuristic solutions relative to the total time to reach the optimal solution. The percentage is calculated as

$$\frac{\text{Time to solve heuristic}}{\text{Time to reach optimal solution}} \times 100\%.$$

Table 16: Heuristic Solution Time

Total Ships	Total Problems	Average MPH Time	Average OSH Time	Average SPH Time
3	80	172.01%	98.33%	112.46%
4	80	101.22%	38.03%	67.49%
5	80	37.13%	12.92%	10.21%
6	80	43.87%	11.29%	22.54%
7	80	24.02%	3.08%	3.21%
8	80	23.94%	1.99%	1.26%
9	80	4.90%	1.35%	1.17%
10	80	3.11%	3.59%	2.72%
11	80	8.11%	2.66%	1.06%
12	80	5.02%	3.22%	2.63%
13	80	5.64%	3.63%	2.58%
14	30	3.46%	1.11%	1.51%
15	10	2.60%	0.73%	1.44%
16	10	1.58%	0.31%	0.35%
17	10	1.60%	1.40%	0.61%
18	10	1.85%	1.19%	0.65%

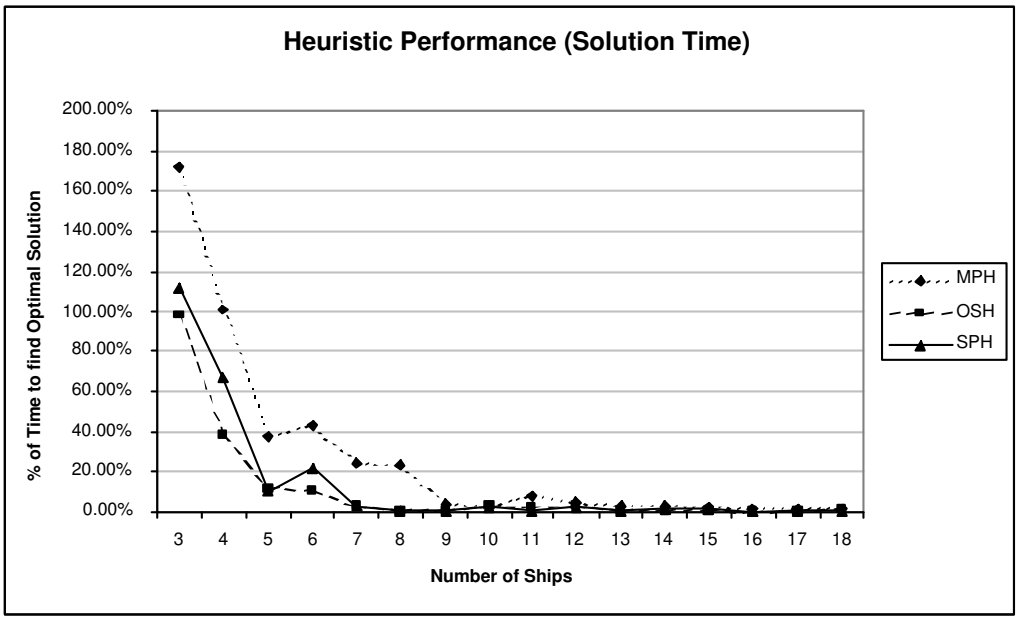

Figure 12: Graph for Heuristic Solution Time

For very small problems, i.e. problems with 3 or 4 ships, we can see that it takes more time to solve the heuristic than that to solve the problem optimally. But as the problem gets larger, we can see that the time to solve the heuristics is getting much smaller than the time to solve the problem optimally, while still maintaining the profit of at least 80% of the optimal.

From Figure 12 we can examine how the proportion of time spent in solving each heuristic compared to that in finding the optimal solution. As the number of ships grows, we need more time to find the optimal solution; but since the time to solve the problem optimally grows exponentially much quicker than the time to solve the heuristics, we may see the proportion of time to solve the heuristics is decreasing as the problem gets bigger. Here we may also see that MPH performs slightly worse than the other two heuristics.

To see how the time to solve the problem grows exponentially, the reader is referred to Figure 13. We see that as the problem becomes harder, the time to solve the problem grows exponentially as expected. To see how the time to solve the heuristics grows, please refer to Figure 14. We only include the solution time to find optimal solutions for small problems due to scaling issues. As we expect, the time to solve the heuristics is much smaller than the time to find the optimal solutions. From the same figure we also find out that MPH grows exponentially faster than OSH and SPH. In conclusion, based on the facts we observed from experiments, we can say that OSH and SPH are more stable in finding solutions to our problems. They also experimentally reach better profits, closer to the optimal solutions than that reached by MPH.

Recall that the time to compute optimal solutions are CPLEX computational time under its default options. If we compare the computational times of the heuristic methods to the time to find optimal solutions based on full column generation, we would have different time performances. Assuming that optimal solutions found by full column generation are obtained in one order of magnitude faster than optimal

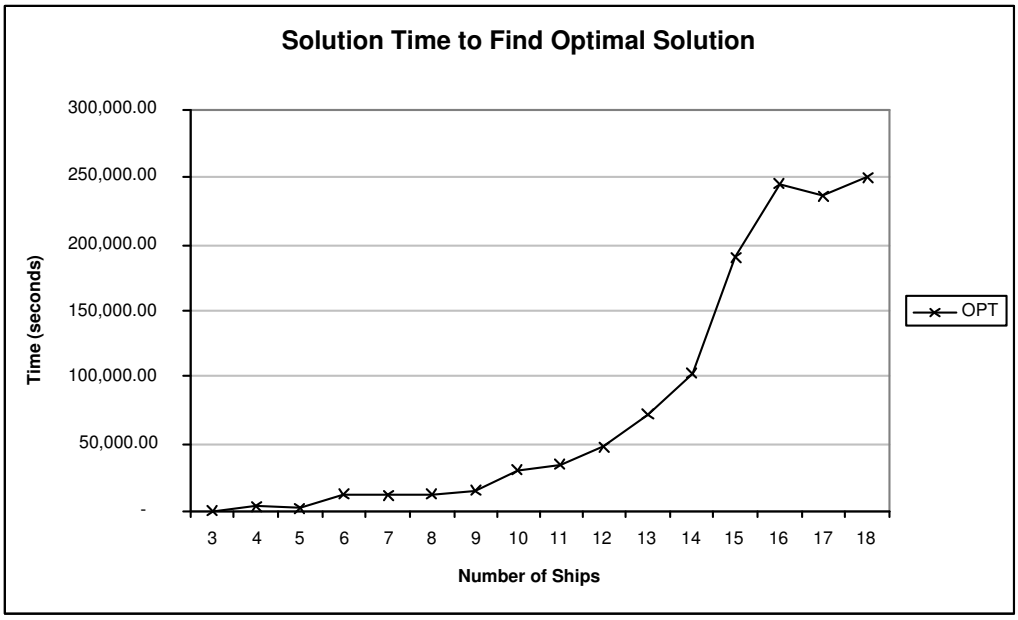

Figure 13: Solution time to find optimal solution

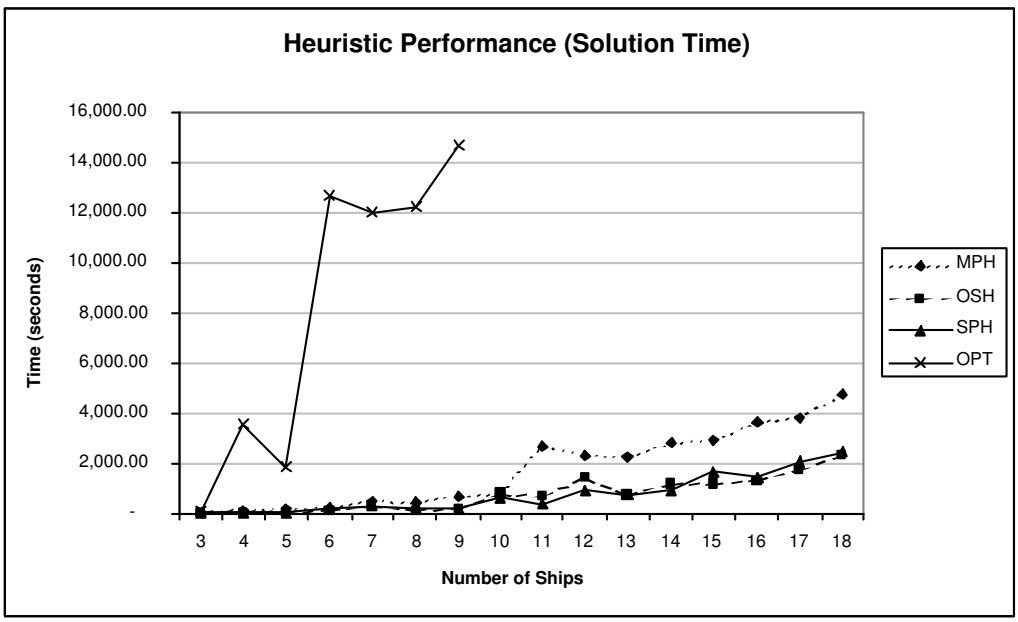

Figure 14: Solution time to find heuristic solutions

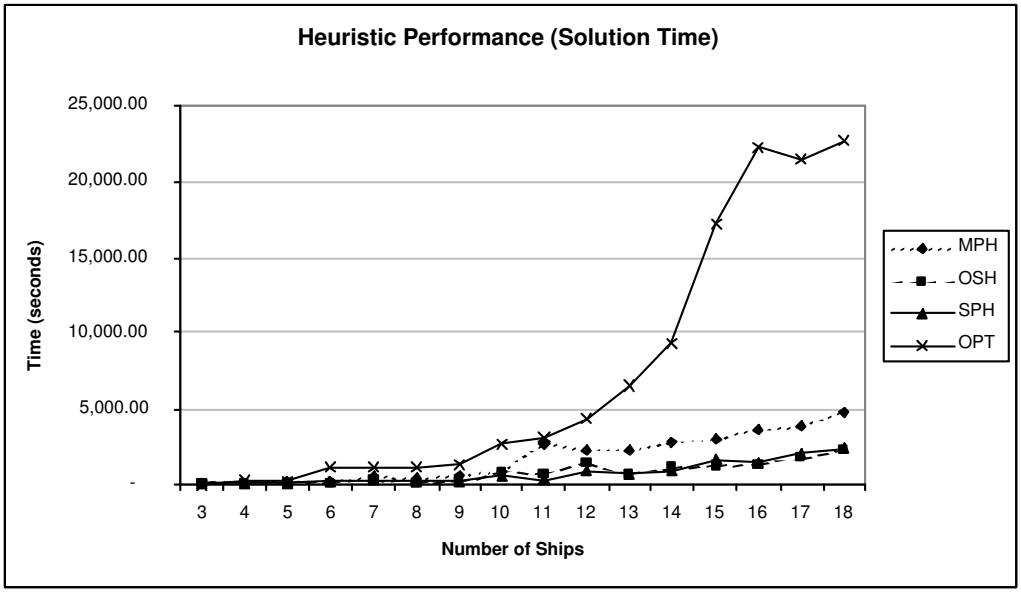

Figure 15: Solution time to find heuristic solutions

solutions found by CPLEX, the graphs in Figures 12 and 13 would have the vertical values differ by an order of magnitude. Also, Figure 14 will look like Figure 15.

CHAPTER IV

UPPER BOUNDING PROBLEM

In this chapter we construct an upper bounding problem to our multi-ship problem. First we explain how to construct an upper bounding problem to the one-ship problem and generalize the idea to derive one for the multi-ship problem. Recall the compact notation (21) for our problem as shown in Section 2.5,

$$\begin{array}{rccccccc}
\text{Maximize} & c_x^1\, x_1 & + & c_x^2\, x_2 & + & c_y\, y & & \\
\text{s.t.} & A_x^1\, x_1 & + & A_x^2\, x_2 & + & A_y\, y & \leq & a \\
& D_x^1\, x_1 & + & D_x^2\, x_2 & & & \leq & d \\
& E\, x_1 & & & & & \leq & e \\
& & & F\, x_2 & & & \leq & f \\
& & & x_1, x_2 \in \{0,1\},\ y \geq 0 & & & &
\end{array}$$

where (c_x^1, c_x^2, c_y) is the objective coefficient row vector, A_x^1, A_x^2, A_y, D_x^1, D_x^2, E and F are real matrices, (a^T, d^T, e^T, f^T) is the right hand side column vector, and x_1, x_2 and y are column vectors, where

$$\begin{aligned}
x &:= \langle X_{isk}, Z_{ilsk}, Y_{js}, XP_{jsk}, XD_{jsk}, XC_{jsk} \rangle \\
y &:= \langle TT_{sk}, T_{sk} \rangle
\end{aligned}$$

range over $j \in \mathcal{L} \cup \mathcal{U},\ i, l \in \mathcal{I},\ s \in \mathcal{S},\ k \in \mathcal{K}_s$. We use the notation $\langle \cdot \rangle$ to denote the column vector with components ranging through the specified indices.

4.1 Deriving an upper bound to the one-ship problem

First, we will focus on one-ship polyhedra. To simplify our notation, let $x^T = (x_1^t,\ x_2^t)$, $c_x = (c_x^1,\ c_x^2)$, $b^T = (d^T, e^T, f^T)^T$, $A_x = (A_x^1,\ A_x^2)$ and

$$B = \begin{bmatrix} D_x^1 & D_x^2 \\ E & 0 \\ 0 & F \end{bmatrix}$$

and we may rewrite our problem as

$$\begin{array}{rrcrcl} \text{Maximize} & c_x x & + & c_y y & & \\ \text{s.t.} & A_x x & + & A_y y & \leq & a \\ & Bx & & & \leq & b \\ & x_1, x_2 \in \{0,1\},\ 0 \leq y \leq H, & & & & \end{array}$$

where H denotes the time horizon. Furthermore, let

$$\begin{aligned} \mathcal{X} &:= \{x \mid Bx \leq b,\ x \in \{0,1\}\} \\ \mathcal{Y} &:= \{y \mid 0 \leq y \leq H\} \end{aligned}$$

so we can write our one-ship problem in a compact form as follows

$$\max_{x \in \mathcal{X}, y \in \mathcal{Y}} \Big\{ c_x x + c_y y \mid A_x x + A_y y \leq a \Big\}. \tag{22}$$

4.1.1 Upper bound

Benders' algorithm has often been proposed to decompose MILP problems as stated in a compact form (22). The decomposition of a problem in the form of (22) consists of an integer programming master problem involving the x variables, and a linear programming subproblem involving the y variables. For fixed binary vector $\overline{x}$, (22) reduces to the linear programming subproblem

$$v(\overline{x}) = \max_{y \in \mathcal{Y}} \Big\{ c_y y \mid A_y y \leq a - A_x \overline{x} \Big\}. \tag{23}$$

We then consider the dual of (23),

$$v(\overline{x}) = \min_{p \in \mathcal{P}} \Big\{ u_p^T (a - A_x \overline{x}) \Big\} \tag{24}$$

where $\{u_p | p \in \mathcal{P}\}$ is the set of extreme points of the dual feasible region, which does not depend on the value of $\overline{x}$, and $\mathcal{P}$ denotes the corresponding index set. Hence, our original problem (22) can be written as a nonlinear integer programming problem

$$\max_{x \in \mathcal{X}} \Big\{ c_x x + v(x) \Big\}.$$

The following inequalities show a step-by-step derivation of an upper-bounding problem

$$\begin{aligned}
& \max_{x \in \mathcal{X}, y \in \mathcal{Y}} \Big\{ c_x x + c_y y \mid A_x x + A_y y \leq a \Big\} \\
= \; & \max_{x \in \mathcal{X}} \Big\{ c_x x + \max_{y \in \mathcal{Y}} \Big\{ c_y y \mid A_y y \leq a - A_x x \Big\} \Big\} && (a) \\
= \; & \max_{x \in \mathcal{X}} \Big\{ c_x x + \min_{u \in \mathcal{V}} \Big\{ u^T (a - A_x x) \mid u^T A_y \geq c_y \Big\} \Big\} && (b) \\
= \; & \max_{x \in \mathcal{X}} \Big\{ c_x x + \min_{p \in \mathcal{P}} \Big\{ u_p^T (a - A_x x) \Big\} \Big\} && (c) \\
= \; & \max_{x \in \mathcal{X}} \Big\{ \min_{p \in \mathcal{P}} \Big\{ c_x x + u_p^T (a - A_x x) \Big\} \Big\} && (d) \\
\leq \; & \min_{p \in \mathcal{P}} \Big\{ \max_{x \in \mathcal{X}} \Big\{ c_x x + u_p^T (b - A_x x) \Big\} \Big\} && (e) \\
= \; & \min_{p \in \mathcal{P}} \Big\{ \max_{x \in \mathcal{X}} \Big\{ u_p^T b + (c_x - u_p^T A_x) \; x \Big\} \Big\} && (f) \\
\leq \; & \min_{p \in \mathcal{P}'} \Big\{ \max_{x \in \mathcal{X}} \Big\{ u_p^T b + (c_x - u_p^T A_x) \; x \Big\} \Big\} && (g) \\
= \; & \min_{p \in \mathcal{P}'} \Big\{ u_p^T b + \max_{x \in \mathcal{X}} \Big\{ (c_x - u_p^T A_x) \; x \Big\} \Big\} && (h)
\end{aligned}$$

where $\mathcal{P}' \subset \mathcal{P}$ is the index set of selected dual extreme points that we obtain by

exploring the special structure in the matrix of the original problem. We will describe the details on how we construct the set $\mathcal{P}'$ in a later section.

The derivation we showed above shows how to go from (a) to (d). We use the min-max theorem (e.g., Danskin [19]) to obtain (e) from (d). The theorem basically says that the minimum among the maximums is always greater than or equal to the maximum among the minimums. Form (f) is just a rearrangement of (e). Finally, the inequality between (f) and (g) holds because we only consider *some* dual extreme points as opposed to *all* possible dual extreme points. The idea is that, if we can choose some "good" dual extreme points, hopefully we can find a "good" upper bound.

Next, we will exploit the special structure of our one-ship problem that will be used later to find some dual extreme points to define a set $\mathcal{P}'$.

4.1.2 Primal and dual pair in one-ship problem

Let us take a closer look at our one-ship problem (23). Stated using the original variables, for each ship $s \in \mathcal{S}$, the ship-s problem can be written as finding the cost of chartering the ship s from time zero until the ship s finishes its operation. Or, we are trying to minimize TCC_s times T_{sK_s}. It is equivalent to $-TCC_s$ times the optimal value of

$$
\begin{array}{lrcll}
\max & T_{sK_s} & & & \\
\text{s.t.} & T_{sk} & \leq & \alpha_{jsk}(x_s), \quad j \in \mathcal{U}, k \in \mathcal{K}_s \backslash \{K_s\} & (\rho_{jsk}) \\
& -T_{s(k+1)} & \leq & \beta_{jsk}(x_s), \quad j \in \mathcal{U}, k \in \mathcal{K}_s \backslash \{K_s\} & (\pi_{jsk}) \\
& T_{sk} - T_{s(k+1)} & \leq & \gamma_{sk}(x_s), \quad k \in \mathcal{K}_s \backslash \{K_s\} & (\sigma_{jsk}) \\
& T_{s0} & = & t_{s0} & (\nu) \\
& T_{sk} & \geq & 0 \qquad k \in \mathcal{K}_s \backslash \{0\} &
\end{array}
\tag{25}
$$

where based on (15), (16), and (19),

$$\begin{aligned}
\alpha_{jsk}(x_s) &= M-(M-LPT_j+\tfrac{1}{2}T_{adm})XP_{jsk} &&\geq 0\\
\beta_{jsk}(x_s) &= -(EPT_j+\tfrac{1}{2}T_{adm})XP_{jsk}-\tfrac{V_jXP_{jsk}}{LR_j}-\textstyle\sum_i\sum_l \frac{\mathrm{Dis}_{il}Z_{ilsk}}{24v_s} &&\leq 0\\
\gamma_{sk}(x_s) &= -T_{adm}(1-X_{0sk})-\textstyle\sum_j \frac{V_jXD_{jsk}}{DR_j}-\sum_j \frac{V_jXP_{jsk}}{LR_j}-\sum_i\sum_l \frac{\mathrm{Dis}_{il}Z_{ilsk}}{24v_s} &&\leq 0.
\end{aligned}$$

and dual variables associated with each constraint are shown in the bracket. Note that we have eliminated variables TT_{sk} by using a substitution according to (72).

The dual problem of (25) is to minimize

$$t_{s0}\ \nu+\sum_{j\in\mathcal{U}}\sum_{k=0}^{K_s-1}\alpha_{jsk}(x_s)\ \rho_{jsk}+\sum_{j\in\mathcal{U}}\sum_{k=0}^{K_s-1}\beta_{jsk}(x_s)\ \pi_{jsk}+\sum_{k=0}^{K_s-1}\gamma_{jsk}(x_s)\ \sigma_{jsk}$$

$$\begin{aligned}
\text{s.t.}\quad v+\textstyle\sum_{j\in\mathcal{U}}\rho_{js0} \qquad & +\sigma_{s0} && = 0 && (T_{s0})\\
\textstyle\sum_{j\in\mathcal{U}}\rho_{jsk} \quad & -\textstyle\sum_{j\in\mathcal{U}}\pi_{js(k-1)} \quad +\sigma_{sk}-\sigma_{s(k-1)} && \geq 0,\ k\in\mathcal{K}_s\backslash\{0,K_s\} && (T_{sk})\\
& -\textstyle\sum_{j\in\mathcal{U}}\pi_{js(K_s-1)} \quad -\sigma_{s(K_s-1)} && \geq -TCC_s && (T_{sK_s})\\
& \nu \text{ unrestricted},\ \rho_{jsk},\pi_{jsk},\sigma_{sk}\geq 0,\ j\in\mathcal{U},k\in\mathcal{K}_s
\end{aligned}$$

4.1.3 Analyzing primal

In our primal problem (25), for each $s\in\mathcal{S}$ the objective is to maximize $-TCC_s\ T_{sK_s}$. Hence it is our intention to make the value of T_{sK_s} as small as possible. For T_{sK_s}, we have the relations,

$$\begin{aligned}
-T_{sK_s} &\leq \beta_{js(K_s-1)}(x_s) && j\in\mathcal{U}\\
T_{sK_s} &\geq T_{s(K_s-1)}-\gamma_{s(K_s-1)}(x_s)
\end{aligned}$$

so we may derive the following lower bound for T_{sK_s}

$$T_{sK_s}\ \geq\ \max\Big\{T_{s(K_s-1)}-\gamma_{s(K_s-1)}(x_s),\ \max_{j\in\mathcal{U}}\big\{-\beta_{js(K_s-1)}(x_s)\big\}\Big\} \tag{26}$$

Since our problem is a maximization problem, in the optimal solution T_{sK_s} will be forced to equal its lower bound, which is achievable. But this lower bound depends

on the value of $T_{s(K_s-1)}$. Similarly, we can derive the lower bound for $T_{s(K_s-1)}$ as

$$T_{s(K_s-1)} \geq \max\Big\{T_{s(K_s-2)} - \gamma_{s(K_s-2)}(x_s),\ \max_{j\in\mathcal{U}}\big\{-\beta_{js(K_s-2)}(x_s)\big\}\Big\} \tag{27}$$

We may substitute (27) in (26) and obtain

$$\begin{aligned} T_{sK_s} \geq \max\Big\{ & T_{s(K_s-2)} - \gamma_{s(K_s-2)}(x_s) - \gamma_{s(K_s-1)}(x_s), \\ & -\gamma_{s(K_s-1)}(x_s) + \max_{j\in\mathcal{U}}\big\{-\beta_{js(K_s-2)}(x_s)\big\}, \\ & \max_{j\in\mathcal{U}}\big\{-\beta_{js(K_s-1)}(x_s)\big\}\Big\}. \end{aligned}$$

We can continue doing this until T_{s0}, and derive a closed form expression for the lower bound of T_{sK_s} which depends on the value of x_s.

$$\begin{aligned} T_{s(K_s-2)} &\geq \max\Big\{T_{s(K_s-3)} - \gamma_{s(K_s-3)}(x_s),\ \max_{j\in\mathcal{U}}\big\{-\beta_{js(K_s-3)}(x_s)\big\}\Big\} \\ T_{s(K_s-3)} &\geq \max\Big\{T_{s(K_s-4)} - \gamma_{s(K_s-4)}(x_s),\ \max_{j\in\mathcal{U}}\big\{-\beta_{js(K_s-4)}(x_s)\big\}\Big\} \\ &\vdots \\ T_{s2} &\geq \max\Big\{T_{s1} - \gamma_{s1}(x_s),\ \max_{j\in\mathcal{U}}\big\{-\beta_{js1}(x_s)\big\}\Big\} \\ T_{s1} &\geq \max\Big\{t_{s0} - \gamma_{s1}(x_s),\ \max_{j\in\mathcal{U}}\big\{-\beta_{js0}(x_s)\big\}\Big\} \end{aligned}$$

And we may derive the closed form for the lower bound of T_{sK_s}. Again, note that we want the value of T_{sK_s} to be as small as possible. Therefore, the optimal value $T^{\star}_{sK_s}$ will be equal to

$$\begin{aligned} T^{\star}_{sK_s} = \max\Big\{ & t_{s0} - \textstyle\sum_{k=0}^{K_s-1}\gamma_{sk}(x_s), \\ & -\textstyle\sum_{k=1}^{K_s-1}\gamma_{sk}(x_s) + \max_{j\in\mathcal{U}}\big\{-\beta_{js0}(x_s)\big\}, \\ & -\textstyle\sum_{k=2}^{K_s-1}\gamma_{sk}(x_s) + \max_{j\in\mathcal{U}}\big\{-\beta_{js1}(x_s)\big\}, \\ & \cdots \\ & -\gamma_{s(K_s-1)}(x_s) + \max_{j\in\mathcal{U}}\big\{-\beta_{js(K_s-2)}(x_s)\big\}, \\ & \max_{j\in\mathcal{U}}\big\{-\beta_{js(K_s-1)}(x_s)\big\}\Big\}, \end{aligned} \tag{28}$$

and the optimal objective value of our primal problem equals $-TCC_s\ T^{\star}_{sK_s}$.

4.1.4 Analyzing dual

With the closed form (28) for the optimal T_{sK_s} value in hand, we compare it to the optimal dual objective function which is obtained by minimizing

$$t_{s0}\ \nu + \sum_{j\in\mathcal{U}}\sum_{k=0}^{K_s-1} \alpha_{jsk}(x_s)\ \rho_{jsk} + \sum_{j\in\mathcal{U}}\sum_{k=0}^{K_s-1} \beta_{jsk}(x_s)\ \pi_{jsk} + \sum_{k=0}^{K_s-1} \gamma_{sk}(x_s)\ \sigma_{jsk}. \tag{29}$$

By analyzing these two forms, we may notice that the closed form (28) of optimal T_{sK_s} actually suggests some possible $\{-1, 0, 1\}$ combinations of $\{\nu, \rho_{jsk}, \pi_{jsk}, \sigma_{jsk}\}$, and we claim that all of these combinations are feasible to the dual problem (33). First, let us take a look at the first term of $T^{\star}_{sK_s}$ in (28), which is

$$t_{s0} - \sum_{k=0}^{K_s-1} \gamma_{sk}(x_s).$$

By comparing this term to the objective term of the dual (29), we may deduce that the following combination is feasible for the dual problem,

$$\begin{aligned} \nu &= 1 \\ \rho_{jsk} &= 0 \quad j \in \mathcal{U},\ k \in \mathcal{K}_s\backslash\{K_s\} \\ \pi_{jsk} &= 0 \quad j \in \mathcal{U},\ k \in \mathcal{K}_s\backslash\{K_s\} \\ \sigma_{sk} &= -1 \quad k \in \mathcal{K}_s\backslash\{K_s\}. \end{aligned} \tag{30}$$

This can be verified by substituting (30) in the dual system. Similarly, from the second term of (28)

$$-\sum_{k=1}^{K_s-1} \gamma_{sk}(x_s) + \max_{j\ \in\ \mathcal{U}} - \beta_{js0}(x_s),$$

for each ship s and every fixed $j^{\star} \in \mathcal{U}$ we have a dual feasible point,

$$
\begin{aligned}
\nu &= 0 \\
\rho_{jsk} &= 0 \quad j \in \mathcal{U},\ k \in \mathcal{K}_s \backslash \{K_s\} \\
\pi_{j^\star sk} &= 1 \quad k \in \mathcal{K}_s \backslash \{K_s\} \\
\pi_{jsk} &= 0 \quad j \in \mathcal{U} \backslash \{j^\star\}, k \in \mathcal{K}_s \backslash \{K_s\} \\
\sigma_{sk} &= -1 \quad k \in \{1, 2, ..., K_s\}.
\end{aligned} \tag{31}
$$

Note that from the second term of (28) alone we have as many as $|\mathcal{U}|$ dual feasible points. Again, the feasibility of these dual points can be easily checked by direct computation of the dual system. Similarly, from the third term of (28) to the last term of (28), we have as many as $|\mathcal{U}|$ dual feasible points each. This leads to a total number of $(1 + K_s \times |\mathcal{U}|)$ dual feasible points that we may obtain from (28) for each ship $s \in \mathcal{S}$. These feasible points define the set $\mathcal{P}'$ and are used in the upper-bounding problem.

Intuitively, each of these feasible points tells us that the minimum operating time $T^\star_{sK_s}$ for ship-s equals the sum of travel time to reach the first destination, all pickup, download, and port administrative times in all legs, and the travel time between ports, with one of the following cases happening:

- there is no waiting time in all legs, the arrival of the ship at the port in each leg lies within the pickup and download time-windows of the cargos the ship has to serve in the respective port (this is the first term in (28));

- there is a waiting time at the port visited in leg-0 because the ship arrives earlier than the pickup time-windows for some cargos being picked-up in leg-0, but there is no delay in the following legs (this is the second term in (28));

- there is a waiting time at the port visited in leg-1 for the same reason as above, but there is no delay in the following legs (here it does not mean that there is no delay in leg-0, but it does not affect the total operational time of the ship since there is a delay in leg-1 anyway);

- similar cases where a delay happens in leg-2 and no delay in the next legs, in leg-3 and no delay in the next legs, etc.

4.1.5 Example

Now we will implement the foregoing ideas to our 10-ship problem. Upper-bounds for all ten one-ship problems (using the method described in Section 4.1.1) are presented in Table 17. We also include the upper bounds from the LP-relaxation for comparison.

Table 17: Upper bound comparison for one-ship problems

Ship	Optimal Solution	LP Relaxation	LP Relaxation (% from Optimal)	Upper Bound (proposed method)	UB Performance (% from Optimal)
1	162117	530349	327.14	225611	139.17
2	195488	673143	344.34	233520	119.45
3	145147	614706	423.51	285975	197.02
4	202064	592275	293.11	295955	146.47
5	174627	423907	242.75	201550	115.42
6	141450	743762	525.81	171295	121.10
7	120718	526519	436.16	133352	110.47
8	54037	598016	1,106.68	92112	170.46
9	164257	499902	304.34	180507	109.89
10	216801	634064	292.46	277267	127.89

The results shown in Table 17 tell us that the upper-bound performance ranges from about 110% to almost 200% of the optimal profit. To see how good the upper bound values are, we include the value of the LP relaxation in the table for us to compare with. We may see that the LP relaxation gives us upper bound values that range from 242.75% to $1,106.68\%$ of the optimal solution while our upper bound values range from 109.89% to 197.02% of the optimal. This leads into a reduction on the upper bound values to a range of $15-50\%$ from the LP relaxation. We will now apply the idea to derive an upper bound for the multi-ship problem.

4.2 Deriving an upper bound for the multi-ship problem

Recall that our original multi-ship problem can be decomposed into as many as $|\mathcal{S}|$ one-ship problems if we take care of the coupling constraint carefully. The only

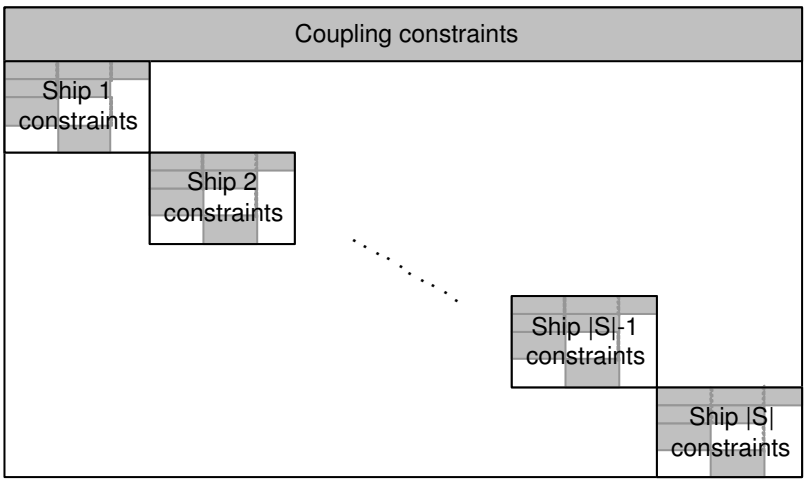

Figure 16: Structure defined by multi-ship polyhedron.

coupling constraint that we have is constraint (71) for as many as $|\mathcal{U}|$ such constraints. The general structure for the multi-ship problem is depicted in Figure 4.2.

Let us rewrite our multi-ship problem as follows:

$$\begin{array}{rrcccl} \text{Maximize} & \sum_{s\in\mathcal{S}} & c_x^s\ x_s & + & c_y^s\ y_s & \\ \text{s.t.} & & H\ x & & \leq & h \\ & & A_x^s\ x_s & + & A_y^s\ y_s \leq & a_s, \quad s \in \mathcal{S} \\ & & D^s\ x_s & & \leq & d_s, \quad s \in \mathcal{S} \\ & & G^s\ x_s & & \leq & g_s, \quad s \in \mathcal{S} \end{array}$$

$$x = \langle x_1, x_2, ..., x_{|\mathcal{S}|} \rangle \in \{0,1\},\ y_s \geq 0\ \forall s \in \mathcal{S}$$

where for each $s \in \mathcal{S}$, (c_x^s, c_y^s) is the objective coefficient row vector, A_x^s, A_y^s, D^s, G^s are real matrices, $(a_s^T, d_s^T, g_s^T)^T$ is the right hand side column vector, and x_s and y_s are column vectors, where

$$\begin{aligned} x_s &:= \langle X_{isk}, Z_{ilsk}, Y_{js}, XP_{jsk}, XD_{jsk}, XC_{jsk} \rangle \\ y_s &:= \langle T_{sk} \rangle \end{aligned}$$

range over $j \in \mathcal{L} \cup \mathcal{U},\ i,l \in \mathcal{I},\ s \in \mathcal{S},\ k \in \mathcal{K}_s$. Note that the coupling constraint that we have is $Hx \leq h$, which represents constraint (71) in our formulation. The variables X_{isk} and Y_{js} are pure binary variables and the remaining variables in x_s are

forced to be binaries by these two pure binary variables. The components of y_s are continuous. The matrix G^s is given by

$$G^s = \begin{bmatrix} E^s & 0 \\ 0 & F^s \end{bmatrix}$$

where E and F are as defined in Section 2.5. We further observe that matrix G is totally-unimodular since submatrices E and F are totally-unimodular by Lemma 2.5.1. Also note that by using the relation reflected in equation (14) we eliminate variable TT_{sk} from the system for all $s \in \mathcal{S}, k \in K_s$.

4.2.1 Upper-bound for the multi-ship problem

We can also write our multi-ship problem as

$$\begin{array}{rrrcrcl} (IP) \quad Z_{IP} = \text{Maximize} & c_x^T\ x & + & c_y^T\ y & & \\ \text{s.t.} & H\ x & & & \leq & h \\ & A_x\ x & + & A_y\ y & \leq & a \\ & D\ x & & & \leq & d \\ & G\ x & & & \leq & g \\ & x \in \{0,1\},\ y \geq 0 & & & & \end{array}$$

where

$$\begin{aligned} x &:= \langle x_1, x_2, ..., x_{|\mathcal{S}|} \rangle \\ y &:= \langle y_1, y_2, ..., y_{|\mathcal{S}|} \rangle \\ c_x &:= \langle c_x^1, c_x^2, ..., c_x^{|\mathcal{S}|} \rangle \\ c_y &:= \langle c_y^1, c_y^2, ..., c_y^{|\mathcal{S}|} \rangle \\ a &:= \langle a_1, a_2, ..., a_{|\mathcal{S}|} \rangle \\ d &:= \langle d_1, d_2, ..., d_{|\mathcal{S}|} \rangle \\ g &:= \langle g_1, g_2, ..., g_{|\mathcal{S}|} \rangle \end{aligned}$$

and

$$A_x = \begin{bmatrix} A_x^1 & & & \\ & A_x^2 & & \\ & & \ddots & \\ & & & A_x^{|\mathcal{S}|} \end{bmatrix}, A_y = \begin{bmatrix} A_y^1 & & & \\ & A_y^2 & & \\ & & \ddots & \\ & & & A_y^{|\mathcal{S}|} \end{bmatrix},$$

$$D = \begin{bmatrix} D^1 & & & \\ & D^2 & & \\ & & \ddots & \\ & & & D^{|\mathcal{S}|} \end{bmatrix}, G = \begin{bmatrix} G^1 & & & \\ & G^2 & & \\ & & \ddots & \\ & & & G^{|\mathcal{S}|} \end{bmatrix}.$$

Let Z_{IP} be the optimal solution of the above problem, we use the same approach as we used for the one-ship problem to derive an upper-bounding problem for (IP).

For a fixed $\overline{x} \in \mathcal{X} := \{x \mid Hx \leq h, Dx \leq d, Gx \leq g, x \text{ binary}\}$, (IP) reduces to the linear programming subproblem

$$\max_{y \geq 0} \Big\{ c_y y \mid A_y y \leq a - A_x \overline{x} \Big\}. \tag{32}$$

We then consider the dual of (32) whose feasible region does not depend on the value of $\overline{x}$,

$$\begin{aligned} & \min_{u \in \mathcal{U}} \Big\{ u^T (a - A_x x) \mid u^T A_y \geq c_y \Big\} \\ = \quad & \min_{p \in \mathcal{P}} \Big\{ u_p^T (a - A_x x) \Big\} \end{aligned} \tag{33}$$

where u_p, $p \in P$ are the set of extreme points of the dual feasible region. Hence, our (IP) problem can be written as an integer programming problem

$$\max_{x\in\mathcal{X}} \left\{ c_x x + \min_{p\in\mathcal{P}} \left\{ u_p^T (a - A_x x) \right\} \right\}$$

$$\leq \min_{p\in\mathcal{P}} \left\{ \max_{x\in\mathcal{X}} \left\{ (c_x - u_p^T A_x)\ x\ + u_p^T a \right\} \right\}$$

$$\leq \min_{p\in\mathcal{P}'} \left\{ \max_{x\in\mathcal{X}} \left\{ (c_x - u_p^T A_x)\ x\ + u_p^T a \right\} \right\}$$

where $\mathcal{P}'$ denotes a subset of $\mathcal{P}$. We have derived an upper-bounding program to our multi-ship problem, which is:

$$(UB) \qquad Z_{UB} = \text{Minimize}_{p\in\mathcal{P}'}\ \ Z_{UB}^p \tag{34}$$

where for each $p \in \mathcal{P}' \subseteq \mathcal{P} = \Big\{ p \mid u_p \text{ is extreme point of polyhedron } \{u \mid u^T A_y \geq c_y\} \Big\}$,

$$\begin{aligned} (UB^p) \quad Z_{UB}^p = u_p^T a \ +\ \text{Maximize} \quad & (c_x - u_p^T A_x)^T x \\ \text{s.t.} \quad & Hx \ \leq\ h \\ & Dx \ \leq\ d \\ & Gx \ \leq\ g \\ & x \in \{0,1\} \end{aligned}$$

4.2.2 Lagrangian relaxation of upper-bounding problem

Since matrix G is totally-unimodular and the elements of vector g are integer, we assume that optimizing over the set $\{x \mid Gx \leq g, x \text{ binary}\}$ can be done efficiently. However, adding constraints and $Hx \leq h$ and $Dx \leq d$ to the problem makes the problem more difficult to solve. We next consider dualizing $Hx \leq h$ while introducing non-negative vectors of multipliers π so that the problem is easier to solve.

For a fixed vector π, we consider the problem

$$\begin{aligned} Z^p_{LR}(\pi) = u_p^T a \ + \ \text{Maximize} \quad & (c_x - u_p^T A_x)^T x + \pi^T (h - Hx) \\ \text{s.t.} \quad & D\ x \leq d, \\ & G\ x \leq g, \\ & x \in \{0,1\}. \end{aligned}$$

and we refer to it as problem $LR^p(\pi)$.

Lemma 4.2.1 *If the problem (IP) in Section 4.2.1 has an optimal solution, for all $\pi \geq 0$, we have the following relationship:*

$$Z_{IP} \leq Z_{UB} = min_{p \in \mathcal{P}'}\ Z^p_{UB} \leq min_{p \in \mathcal{P}'}\ Z^p_{LR}(\pi). \tag{35}$$

The proof is omitted since the derivation of the first inequality was explained in Section 4.2.1 and the rest follows by definition. The problem $LR^p(\pi)$ provides an upper-bound on the integer programming problem (IP) for all $\pi \geq 0$. Of course, we want to have the tightest of such bound. Thus, our objective is to find an optimal $\pi^* \geq 0$ that yields $Z^p_{LR}(\pi), p \in \mathcal{P}'$, the and Dual Accent method below can be applied to obtain such π^*.

4.2.3 Dual Ascent Method

Let $\mathcal{V}$ be the set of all extreme points of the polyhedron defined by $\{x\ |Dx \leq d,\ Gx \leq g, x$ binary$\}$ and let $\mathcal{I}$ be the index set for points in $\mathcal{V}$. As we may see, $Z^p_{LR}(\pi)$ is a piecewise linear convex function of π. For every point $x^i \in \mathcal{V}$, the vector $\sum_{i \in \mathcal{I}}\ \alpha_i\ (h - Hx^i)$, such that $\sum_{i \in \mathcal{I}} \alpha_i = 1$ and $\alpha_i \geq 0$ for all $i \in \mathcal{I}$, and all of its convex combinations are in the sub-differential of LR^p at π, which is denoted by $\partial LR^p(\pi)$.

By using the basic property of convex functions, we verify optimality of the current π^k at each iteration k. If the current π^k is not optimal, we derive the improving direction, and choose the step size along the improving direction.

The multiplier π^* is optimal to our problem $LR^p(\pi)$ if and only if the Lagrange multiplier vector satisfies the first order optimality condition, or that the vector zero is in the sub-differential set of $LR^p(\pi)$ at π_H; i.e., $0 \in \partial LR^p(\pi^*)$.

For a given π, the method finds a vertex in $\mathcal{V}$ and checks if $0 \in \partial LR^p(\pi)$. If not, the method will find an improving direction as well as an optimal step size to fix the value of π in the next iteration.

At iteration k, we have a set of extreme points $\mathcal{V}_k \subseteq \mathcal{V}$ that solve $LR^p(\pi^k)$ with associated index $\mathcal{I}_k \subseteq \mathcal{I}$. We want to know if $0 \in \partial LR^p(\pi^k)$. This is the same as checking the existence of α_i such that

$$\sum_{i \in \mathcal{I}} \alpha_i \, [h - Hx^i] = 0, \sum_{i \in \mathcal{I}} \alpha_i \; = 1, \quad \text{and } \alpha_i \geq 0, \text{ for all } i \in \mathcal{I}_k.$$

To check this, we construct the Phase I linear problem as follow

$$\begin{array}{lll} (PH1_k^p) & \text{Min} & \sum_{j=1}^{m} s_j \; + \; s_{m+1} \\ & \text{s.t.} & (Hx^1)\, \alpha_1 + \ldots + (Hx^{|\mathcal{I}_k|})\, \alpha_{|\mathcal{I}_k|} \; + I^m s \;\; = \;\; h \\ & & \alpha_1 + \ldots + \alpha_{|\mathcal{I}_k|} \; + s_{m+1} \;\; = \;\; 1 \\ & & \alpha_i \geq 0, \; \forall i \in \{1, 2, ..., |\mathcal{I}_k|\} \\ & & s_j \geq 0, \; \forall j \in \{1, 2, ..., m+1\} \end{array}$$

where m is the number of rows in matrix H and I^m is the identity matrix of order m, vector of variables $s = [s_1, s_2, ..., s_m]$ and variable s_{m+1} denote artificials.

If the solution of (PH_k^p) is 0 then our current π^k is optimal. Otherwise, the dual variables from the dual problem of (PH_k^p) will give us an improving direction. Let (ρ^k, ρ_0^k) be the corresponding dual variables; i.e., the dual of (PH_k^p) can be written as

$$\begin{array}{lll} \text{Max} & (\rho^k)^T h + \rho_0^k & \\ \text{s.t.} & (Hx^i)^T \rho^k + \rho_0 \;\; \leq \;\; 0, & \text{for all } i \in \mathcal{I}_k, \\ & \rho^k, \rho_0^k \;\; \leq \;\; 1. & \end{array}$$

If the solution of (PH_k^p) is positive, then we know that the dual also has the same solution, hence

$$(\rho^k)^T h + \rho_0^k > 0,$$

or,

$$-\rho_0^k < (\rho^k)^T h, \tag{36}$$

and we also know that the dual problem is feasible, so

$$(Hx^i)^T \rho^k + \rho_0^k \leq 0, \text{ for all } i \in \mathcal{I}_k,$$

or,

$$(Hx^i)^T \rho^k \leq -\rho_0^k, \text{ for all } i \in \mathcal{I}_k. \tag{37}$$

From (36) and (37) we have that

$$(Hx^i)^T \rho^k < (\rho_H^k)^T h$$

or,

$$(\rho_H^k)^T \ (h - Hx^i) > 0, \text{ for all } i \in \mathcal{I}_k. \tag{38}$$

The condition (38) remains true only for all $i \in \mathcal{I}_k$, but not for all $i \in \mathcal{I}$. Therefore, we want to check if the direction ρ_H^k is really an improving direction by checking to see if the directional derivative of LR^p at π^k in the direction ρ^k is positive; i.e., if

$$\lim_{t \to 0} \frac{LR^p(\pi^k + t\rho^k) - LR^p(\pi^k)}{t} \ > \ 0. \tag{39}$$

If ρ^k is really an improving direction, we may update our current value of π^k to $(\pi^k + t\rho^k)$, where the value of t can be determined by the optimal step size. Otherwise, if ρ^k is *not* an improving direction, we will find a new extreme point $x^s \neq x^i, i \ \in \mathcal{I}_k$ as an optimal solution to $LR^p(\pi^k)$. We then update the set $\mathcal{V}_k \leftarrow \mathcal{V}_k \cup \{x^s\}$ and re-solve the phase-one problem. This procedure is valid by a well known result; e.g., Theorem 6.3.4 in [10], which is stated without proof in Proposition 4.2.1.

Proposition 4.2.1 *Suppose* $LR^p(\pi^k + t_s\rho^k) - LR^p(\pi^k) leq 0$ *for* $t_s > 0$ *sufficiently small and* ρ^k *is an optimal dual sub-vector associated with the first constraint of problem* $(PH1_k^p)$. *Let the vertex* x^s *be an optimal solution of* $LR^p(\pi^k + t_s\rho^k)$. *Then* $x^s \in \mathcal{V} \backslash \mathcal{V}_k$.

If the first case happens; i.e., ρ^k *is* an improving direction, the step size t^k can be determined by solving:

$$\text{Max}_{t \geq 0}\ LR^p(\rho^k + t^k \rho^k).$$

After we have the value of t^k, we define $\pi^{k+1} = \pi^k + t^k \rho^k$. Let x^{k+1} be the solution that yields $Z^p_{LR}(\pi)$. We then construct the set $\mathcal{V}_{k+1} = \{x^{k+1}\}$, set $k \leftarrow k+1$ and repeat the iteration.

Furthermore, we can actually decompose $Z^p_{LR}(\pi)$ into as many as $|\mathcal{S}|$ subproblems $LR^p_s(\pi)$

$$Z^p_{LR}(\pi) = u_p^T a \ + \ \pi^T h \ + \ \sum_{s \in \mathcal{S}} Z^{ps}_{LR}(\pi),$$

where $Z^{ps}_{LR}(\pi)$ is the optimal value of problem $LR^p_s(\pi)$ which is defined as follows

$$\begin{aligned} Z^{ps}_{LR}(\pi) = \text{Maximize} \quad & \left[c^s_x - (u^s_p)^T A^s_x - (\pi^s)^T H^s\right]^T x_s \\ \text{s.t.} \quad & D^s\ x_s \leq d^s, \\ & G^s\ x_s \leq g^s, \\ & x_s \in \{0,1\}. \end{aligned}$$

4.3 Computational Results

We apply the foregoing ideas to calculate upper bounds on test case problems that were randomly generated to show the performance of the heuristics we explained in Chapter 3. However, instead of solving the integer program $LR^p_s(\pi)$, we solve the LP Relaxation of it. The resulting bound is not as strong as $Z^{ps}_{LR}(\pi)$, but it still yields an upper bound to the multi-ship problem. Also, we only choose a point $p \in \mathcal{P}'$ that is the first term of the closed form (28). Table 18 shows the performance of the upper bounding problems on the test case problems, measured relative to the optimal solution and to the LP relaxations.

Table 18: Upper bound performance

# Ships	Number of problems	UB Solution Performance (Average % of LP Relaxation)	UB Time Performance (Average % of time to solve LP)
3	80	68.79%	27.54%
4	80	64.33%	17.11%
5	80	74.64%	27.04%
6	80	63.21%	16.81%
7	80	68.50%	29.58%
8	80	68.04%	23.49%
9	80	75.62%	18.15%
10	80	69.23%	21.23%
11	80	71.97%	21.86%
12	80	75.00%	22.29%
13	80	75.27%	23.30%
14	30	69.90%	25.51%
15	10	77.46%	26.60%

The upper bound solution value performance is measured as

$$\frac{\text{UB solution value}}{\text{LP relaxation solution value}} \times 100\%,$$

and the upper time solution time performance is measured as

$$\frac{\text{Time to solve UB problem}}{\text{Time to solve LP relaxation}} \times 100\%.$$

The smaller of these two upper bound performance measurement values, the better they are since they represent how good the bound obtained by our upper bounding problem compared to that of the LP relaxation.

For our particular 10-ship problem example, the LP relaxation solution value is $1,994,850$ obtained in 349.88 seconds while our upper bounding problem gives a slightly better solution of $1,804,170$, or about 90% from the LP relaxation, obtained in 218.36 seconds, or about 62% of the time spent in solving the LP relaxation. We also recall that the optimal solution to our 10-ship example is $1,137,154$. Comparing to this value, the LP relaxation and our upper bound gives a value of 75.42% and 58.66% higher than the optimal solution, respectively. In general, the relaxation of the upper bounding problem gives a better bound compared to that provided by the

LP relaxation solution. Moreover, the tighter bound can be obtained in a much faster time.

Another type of upper bound that we can compute is an upper bound provided by one-ship optimal solutions. The sum of one-ship optimal profit for each ship gives another upper bound. Clearly, profit reached by each ship in the final multi-ship schedule is bounded above by the optimal profit reached by one-ship problem. Otherwise, a ship may serve more cargos to earn extra profit, which leads to a contradiction to the fact that the profit is optimal for one-ship problem. For our particular 10-ship problem, for example, the upper bound obtained by summing up one-ship optimal solutions is $1,538,521$, or 35.30% higher than the optimal solution. While this is a better bound to our problem, the bound is obtained at a slightly higher computational time since we have to solve one-ship problem for each of the ship. The total time to solve all one-ship problems is 804.52 seconds, a slightly longer time compared to that to solve LP relaxation of the multi-ship problem.

CHAPTER V

EXTENSIONS TO SOFT TIME-WINDOWS AND INTER-SHIP CARGO-TRANSFERS

One of the advantages of our model is that we have continuous time variables that allow us to embed extensive time aspects. Here we are going to explore two extensions that allow us to demonstrate this capability. We are going to extend the model to reflect soft time-window and cargo-transfer related activities. Both of these extensions involve time aspects. In the soft time-window extension, we allow pickups and downloads to happen outside the predefined time-windows by imposing some penalty terms. Hence we will measure how far away a pickup or download happens outside its original time-windows. This is where the time aspects come into play. In our cargo-transfer extension, we allow a cargo to be picked-up and downloaded by different ships. For example, we want to have a cargo be picked-up by ship A and downloaded by ship B. For this to happen, we require a transfer from A to B, where we require the download from ship to happen before the pickup by ship B. This is where we can utilize the time aspects of our model.

5.1 Soft time-windows

Recall that in scheduling with hard time windows in our sea-cargo routing and scheduling problem, each cargo has a time window within which a pickup can only occur. Now, we are going to relax that by introducing a soft time windows. By replacing hard time windows with soft time windows, the pickup is allowed to begin outside the original pickup time windows with inconvenience cost or penalty cost incurred. This penalty can represent the cost of lost sales, waiting idle in the port, goodwill, etc. The motivation is that by allowing time window violations for some cargos, it

may be possible to generate more profits by serving more cargos, or significantly reduce the overall transportation costs. Soft time windows also reflect more flexibility experienced in practice than hard time windows. It is important to note that soft time windows are the more general case so it includes the hard time windows case as well. To see this, suppose our hard time window for a cargo j is $[EPT_j, LPT_j]$. Extending this time window to $[XEPT_j, XLPT_j]$, where $XEPT_j = EPT_j - \delta_e$ and $XLPT_j = LPT_j + \delta_l$, yields a soft time window. It is clear to see that by setting $\delta_e = \delta_l = 0$ we arrive at our original hard time window case.

5.1.1 Embedding soft time-windows into our model

To extend our problem to accommodate soft time-windows, we may easily adjust constraints (15) and (16) in our original problem by replacing EPT_j and LPT_j with $XEPT_j$ and $XLPT_j$, respectively, where "X" indicates the "extended" time-windows. The adjusted constraints become

$$T_{sk} \leq (XLPT_j - \frac{1}{2}T_{adm})XP_{jsk} + M(1 - XP_{jsk}) \qquad j \in \mathcal{U}, s \in \mathcal{S}, k \in \mathcal{K}_s \backslash \{K_s\} \quad (40)$$

$$T_{s(k+1)} \geq (XEPT_j + \frac{1}{2}T_{adm})XP_{jsk} + \frac{V_j XP_{jsk}}{LR_j} + TT_{sk} \qquad j \in \mathcal{U}, s \in \mathcal{S}, k \in \mathcal{K}_s \backslash \{K_s\} \quad (41)$$

Introducing penalty terms in the objective function

Penalty is introduced in the objective function and is incurred if a cargo is picked up by a ship after its predetermined time window but before its extended (soft) time window expires. Penalties are usually computed daily; i.e., the company is penalized for a fixed amount for each late pick-up day. These penalties are depicted in Figure 17 and given by the penalty function

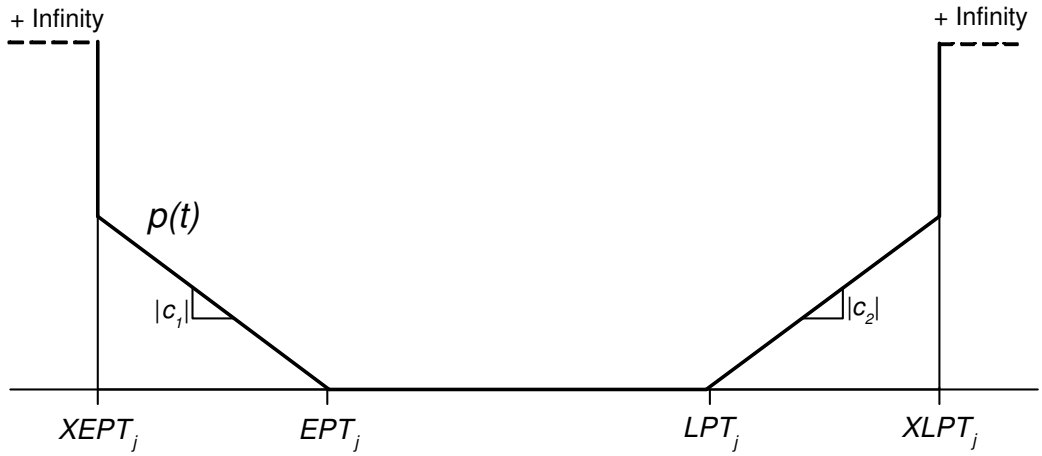

Figure 17: Penalty function for soft time-windows

$$p_j(t) = \begin{cases} +\infty & \text{if } t < XEPT_j \\ |c_1|(EPT_j - t) & \text{if } XEPT_j \leq t < EPT_j \\ 0 & \text{if } EPT_j \leq t \leq LPT_j \\ |c_2|(t - LPT_j) & \text{if } LPT_j < t \leq XLPT_j \\ +\infty & \text{if } t > XLPT_j \end{cases}$$

where t denotes the time when a ship s is ready to pickup cargo j at the end of leg k.

Suppose a cargo j has soft time window $[XEPT_j, XLPT_j]$ and is supposed to be picked up by ship s. The inconvenience/penalty cost is incurred when ship s arrives in either interval $[XEPT_j, EPT_j]$ or $[LPT_j, XLPT_j]$. If the ship arrives in interval $[EPT_j, LPT_j]$, the inconvenience cost is zero and if the ship arrives before $XEPT_j$ or after $XLPT_j$, there is no chance for ship s to serve cargo j.

In this model, the inconvenience cost is denoted as P_{jsk} and is a linear function of the degree of violation. Suppose the time when ship s arrives is T_{sk}. Since the ship must go through an administrative process once arrived at a port, the degree of violation is measured by $EPT_j - (T_{sk} + T_{adm})$ if ship s arrives before EPT_j or $(T_{sk} + T_{adm}) - LPT_j$ if ship s arrives after LPT_j.

Due to the nature of this problem, it does not make sense to incur a penalty cost

when a ship picking up a cargo arrives before the earliest pickup time. Therefore a penalty cost is only introduced when the ship arrives after the latest pickup time. We revise our penalty function as follows:

$$p_j(t) = \begin{cases} +\infty & \text{if } t < XEPT_j \\ 0 & \text{if } XEPT_j \leq t \leq LPT_j \\ \widetilde{c}(t - LPT_j) & \text{if } LPT_j < t \leq XLPT_j,\ \widetilde{c} > 0 \\ +\infty & \text{if } t > XLPT_j \end{cases}$$

where $t = T_{sk} + 0.5T_{adm}$.

Clearly, penalty for cargo j occurs if the pickup of cargo j is performed between the latest pickup time LPT_j and the extended latest pickup time $XLPT_j$. And it only occurs when there is a pickup for cargo j. Hence, for each cargo j, the penalty cost is $P_{jsk} = XP_{jsk}\ p_j(t)$. We may also write it as:

$$P_{jsk} = \widetilde{c}\ XP_{jsk}\ \max\{0,\ T_{sk} + 0.5T_{adm} - LPT_j\}$$

To capture all penalty costs, a new term should be introduced in the original objective function. Let $f(\cdot)$ be the original objective function to be maximized. Then the positive penalty terms are subtracted from the objective.

$$\text{Maximize } f(\cdot) - \sum_{j \in \mathcal{U}} \sum_{s \in \mathcal{S}} \sum_{k \in \mathcal{K}_s} P_{jsk}$$

or equivalently,

$$
\begin{aligned}
& \text{Maximize } f(\cdot) - \textstyle\sum_j \sum_s \sum_k \widetilde{c}\; XP_{jsk} \max\{0,\; T_{sk} + 0.5T_{adm} - LPT_j\} \\
\Leftrightarrow\; & \text{Maximize } f(\cdot) - \textstyle\sum_j \sum_s \sum_k \widetilde{c} \max\{0,\; T_{sk}\; XP_{jsk} + (0.5T_{adm} - LPT_j)\; XP_{jsk}\} \\
\Leftrightarrow\; & \text{Maximize } f(\cdot) - \textstyle\sum_j \sum_s \sum_k \widetilde{c} \min\{0,\; (LPT_j - 0.5T_{adm})\; XP_{jsk} - T_{sk}\; XP_{jsk}\} \\
\Leftrightarrow\; & \text{Maximize } f(\cdot) - \textstyle\sum_j \sum_s \sum_k \widetilde{c}\; XL_{jsk} \\
& \text{s.t. } XL_{jsk} = \min\{0,\; (LPT_j - 0.5T_{adm})\; XP_{jsk} - T_{sk}\; XP_{jsk}\}, \text{for all } j, s, k \\
\Leftrightarrow\; & \text{Maximize } f(\cdot) - \textstyle\sum_j \sum_s \sum_k \widetilde{c}\; XL_{jsk} \\
& \text{s.t.} \quad XL_{jsk} \;\leq\; 0 \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad j \in \mathcal{U}, s \in \mathcal{S}, k \in \mathcal{K}_s \\
& \qquad\; XL_{jsk} \;\leq\; (LPT_j - 0.5T_{adm})\; XP_{jsk} - T_{sk}\; XP_{jsk} \quad j \in \mathcal{U}, s \in \mathcal{S}, k \in \mathcal{K}_s \\
\Leftrightarrow\; & \text{Maximize } f(\cdot) - \textstyle\sum_j \sum_s \sum_k \widetilde{c}\; XL_{jsk} \\
& \text{s.t.} \quad XL_{jsk} \;\leq\; 0 \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad j \in \mathcal{U}, s \in \mathcal{S}, k \in \mathcal{K}_s \\
& \qquad\; XL_{jsk} \;\leq\; (LPT_j - 0.5T_{adm})\; XP_{jsk} - TXP_{jsk} \quad j \in \mathcal{U}, s \in \mathcal{S}, k \in \mathcal{K}_s \\
& \qquad\; TXP_{jsk} \;=\; T_{sk}\; XP_{jsk} \qquad\qquad\qquad\qquad\qquad\qquad j \in \mathcal{U}, s \in \mathcal{S}, k \in \mathcal{K}_s
\end{aligned}
$$

5.1.2 Exact linear relaxation

The foregoing formulation has bilinear terms $TXP_{jsk} = T_{sk}\; XP_{jsk}$ for all $j \in \mathcal{U}, s \in \mathcal{S}, k \in \mathcal{K}_s$. We will construct an exact linear reformulation for each of these bilinear constraints. Noting that XP_{jsk} is 0-1 (implied) integer variables, TXP_{jsk} lie in the interval $[0, H]$, where H denotes the length of the planning horizon. As a result, we arrive at the equivalent reformulation

$$
\begin{aligned}
& \text{Maximize } f(\cdot) - \textstyle\sum_j \sum_s \sum_k c\; XL_{jsk} \\
& \text{s.t.} \quad XL_{jsk} \;\leq\; 0 \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad j \in \mathcal{U}, s \in \mathcal{S}, k \in \mathcal{K}_s \\
& \qquad\; XL_{jsk} \;\leq\; (LPT_j - 0.5T_{adm})\; XP_{jsk} - TXP_{jsk} \quad j \in \mathcal{U}, s \in \mathcal{S}, k \in \mathcal{K}_s \\
& \qquad\; TXP_{jsk} \;\geq\; 0 \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad j \in \mathcal{U}, s \in \mathcal{S}, k \in \mathcal{K}_s \\
& \qquad\; TXP_{jsk} \;\geq\; H\; XP_{jsk} + T_{sk} - H \qquad\qquad\qquad\qquad j \in \mathcal{U}, s \in \mathcal{S}, k \in \mathcal{K}_s \\
& \qquad\; TXP_{jsk} \;\leq\; H\; XP_{jsk} \qquad\qquad\qquad\qquad\qquad\qquad\qquad j \in \mathcal{U}, s \in \mathcal{S}, k \in \mathcal{K}_s \\
& \qquad\; TXP_{jsk} \;\leq\; T_{sk} \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad j \in \mathcal{U}, s \in \mathcal{S}, k \in \mathcal{K}_s
\end{aligned}
$$

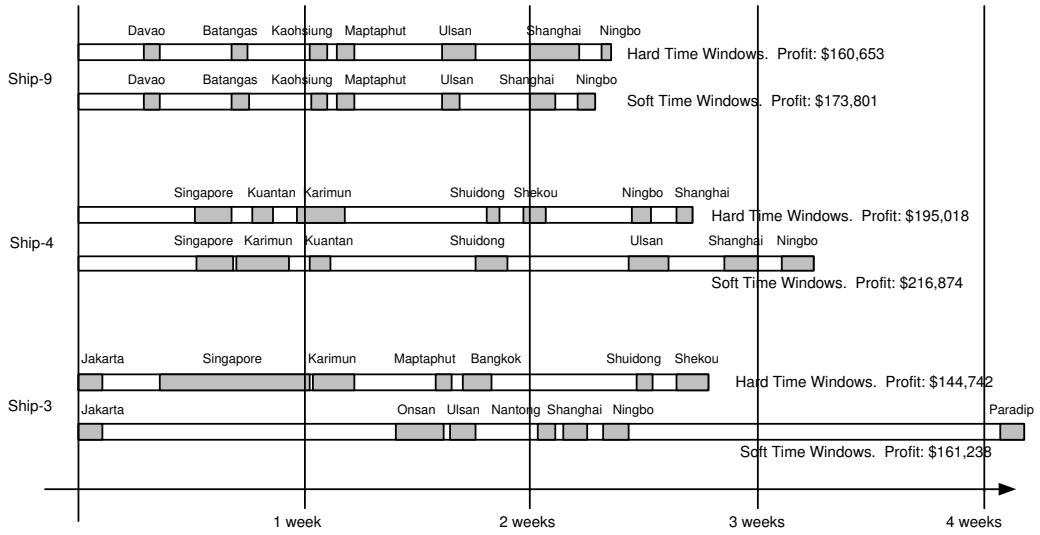

Figure 18: Comparison of Gantt-charts for the optimal one-ship schedules with hard and soft time windows for ships 3,4, and 9.

Consequently, the introduction of soft time-windows increases the size of the problem. Our problem has an additional $(|\mathcal{U}| \times |\mathcal{S}| \times |\mathcal{K}_s|)$ continuous variables and $(4 \times |\mathcal{U}| \times |\mathcal{S}| \times |\mathcal{K}_s|)$ constraints.

5.1.3 Example for one-ship problem

We implement the idea of introducing soft time windows to our 10-ship example. We rerun the one-ship problem for all ships and obtain improvements in three of the 10 ships. Table 19 shows the improvements in the profits gained by ships 3, 4, and 9. The penalty we imposed is 3 percent of the shipping rate discount per day, allowing a maximum lateness of 7 days.

Table 19: Comparison of hard and soft time window (TW) profits for one ship problems

Ship	Cargos onboard	New Cargos	Late-pickup Cargos	Lateness (days)	Hard TW Profit ($)	Soft TW Profit ($)	Percent Gained
3	56	11-15,36,38	36,38	1.98,3.49	144,742	161,328	11.46
4	57	9,11-14,17,19,21-25	11-14	5.22	195,108	216,874	22.58
9	68-74	11-14,38	38	3.45	160,653	173,801	8.18

The example shows that by relaxing the hard time windows into soft ones, we may gain better profit from our routing and scheduling decisions. Moreover, soft time windows are more common in practice.

5.2 Cargo-transfer between ships

In the type of ship cargo scheduling models under investigation, we have not found in the literature any model that allows transloads, or cargo transfers from one ship to another. The transfer can be done in any transload port. This is an important feature that can significantly reduce transportation costs (e.g., time-charter cost and fuel cost). This section describes how to model cargo transfer and embed it into our multi-ship problem.

5.2.1 Embedding cargo-transfer into our model

To include cargo-transfer in our model, we introduce new variables and constraints. To simplify our notation, some assumptions to our model are:

(i) A cargo-transfer can happen between two ships at any port. It cannot happen in any open space in the middle of the ocean. In a hub and spoke system, cargo-transfer can only happen at specified transfer ports. In our model, we can always add some restrictions to allow cargo-transfer happening only at certain ports.

(ii) We limit the maximum number of transfers to be one for each cargo. This avoids a possibility of transferring a cargo over and over again even when it is profitable. We believe that the service level would be downgraded if we allow too many transfers.

(iii) The only transfer cost involved is the port charges to be paid by ships when they berth at a transfer port. A cargo may spend some time waiting at the transfer port to be picked-up by another ship, but we ignore this waiting cost since it usually constitutes only a very small portion of the other costs incurred.

(iv) All onboard cargos carried by a ship at time zero cannot be transferred to another ship. In other words, a ship has to deliver all onboard cargos by itself.

The fundamental change is that when a ship s serves a cargo j, it no longer has to pickup the cargo from its pickup port and carry it all the way to the download port. A ship may only pickup cargo j and download it at any available transfer port $i \in \mathcal{I}$. Or, a ship may also pickup the cargo from any transfer port and finally download it at its download port. Remember that we only allow one transfer for each cargo.

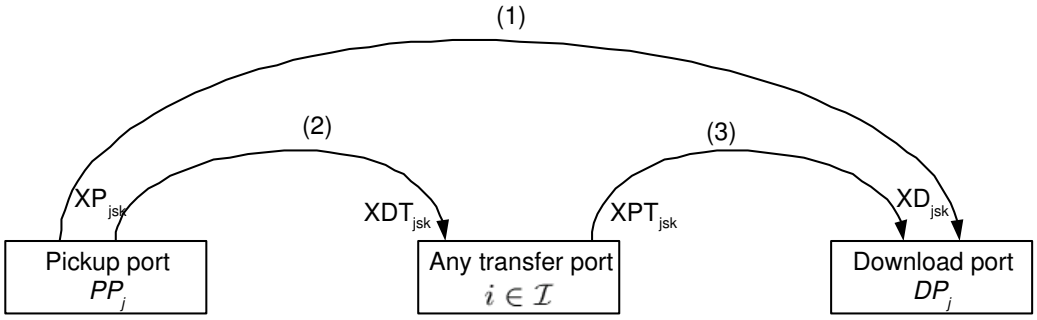

Figure 19: Three ways ship s can serve cargo j when transfers are allowed.

To incorporate this cargo-transfer into the model, in addition to our current pickup variable XP_{jsk} and download variable XD_{jsk}, we introduce new pickup and download variables for cargo-transfer activity, XPT_{jsk} and XDT_{jsk}, respectively

$$XP_{jsk} = \begin{cases} 1 & \text{if ship } s \text{ picks up cargo } j \text{ from the pickup port } PP_j \text{ at the end of leg } k \\ 0 & \text{otherwise} \end{cases}$$

$$XD_{jsk} = \begin{cases} 1 & \text{if ship } s \text{ downloads cargo } j \text{ at the download port } DP_j \text{ at the end of leg } k \\ 0 & \text{otherwise} \end{cases}$$

$$XPT_{jsk} = \begin{cases} 1 & \text{if ship } s \text{ picks up cargo } j \text{ from a transfer port } i \neq PP_j \text{ at the end of leg } k \\ 0 & \text{otherwise} \end{cases}$$

$$XDT_{jsk} = \begin{cases} 1 & \text{if ship } s \text{ downloads cargo } j \text{ at a transfer port } i \neq DP_j \text{ at the end of leg } k \\ 0 & \text{otherwise} \end{cases}$$

As depicted in Figure 19, when a ship serves a cargo, the ship may either (1) pickup the cargo from its pickup port and carry it all the way to the download port, or (2) pickup the cargo from the pickup port and download it at any transfer port, or (3) pickup the cargo from a transfer port and download it at the destination port.

Old constraints modified

Because of the newly introduced variables, some of our current constraints are affected. Those affected constraints are (6), (7), (10), (11), (19), and (13). The rest of the constraints remain the same.

Cargo pickups and downloads

If a ship s serves a cargo j, it has to pick it up once, either from its original pickup port PP_j, or from some transfer port $i \in \mathcal{I}$. If the latter occurs, that means that another ship has already served the cargo and a transfer of cargo j is happening at port i. Similarly, if a ship s serves a cargo j, it has to download it once, either at its destination port DP_j, or at some transfer port $i \in \mathcal{I}$. The following constraints replace our original constraints (6) and (7)

$$\sum_k XP_{jsk} + \sum_k XPT_{jsk} = Y_{js}, \qquad j \in \mathcal{U}, s \in \mathcal{S}, \tag{42}$$

$$\sum_k XD_{jsk} + \sum_k XDT_{jsk} = Y_{js}, \qquad j \in \mathcal{U}, s \in \mathcal{S}. \tag{43}$$

Pickup and download sequence

A pickup has to happen before a download. The following constraint replaces our original constraint (10)

$$\sum_{k>0} k(XD_{jsk} - XPT_{jsk} + XDT_{jsk} - XP_{jsk}) \geq Y_{js}, \qquad j \in \mathcal{U}, s \in \mathcal{S}. \tag{44}$$

Cargo carrying by a ship

To make a ship carry a cargo j from a pickup port to a download port, the following constraint is used to replace constraint (11)

$$XC_{js(k+1)} = XC_{jsk} + XP_{jsk} - XD_{jsk} + XPT_{jsk} - XDT_{jsk}, j \in \mathcal{L} \cup \mathcal{U}, s \in \mathcal{S}, k \in \mathcal{K}_s \backslash \{K_s\}. \tag{45}$$

Time spent at a port

Total service time in a port must exceed the time required for inspections plus the time for discharging all delivery cargos plus the time for loading pickup cargos. The time spent for all transfer activities has to be included here. For the dummy port, we do not have administrative time. The following constraint replaces our original constraint (19)

$$T_{s(k+1)} \geq T_{sk} + T_{adm}(1 - X_{0sk}) + \sum_j \frac{V_j}{DR_j}(XD_{jsk} + XDT_{jsk}) + ... \\ \sum_j \frac{V_j}{LR_j}(XP_{jsk} + XPT_{jsk}) + TT_{sk}, s \in \mathcal{S}, k \in \mathcal{K}_s \backslash \{K_s\}. \tag{46}$$

Limit on number of ships serving a cargo

As we stated in one of the assumptions, a cargo-transfer can happen at most once for each cargo. Since the maximum number of ships serving a cargo is two, constraint (47) replaces constraint (13) in our original formulation

$$\sum_{s} Y_{js} \leq 2, \qquad j \in \mathcal{L} \cup \mathcal{U}. \tag{47}$$

5.2.2 Change in the objective function

In our original formulation, we gained revenues by delivering a cargo from its pickup port to its destination port. This is reflected in the objective function as

$$\sum_{s \in \mathcal{S}} \sum_{j \in \mathcal{L} \cup \mathcal{U}} SR_j \; Y_{js}. \tag{48}$$

But now, since we allow at most two ships to serve a cargo, we may have $\sum_s Y_{js} = 2$ for some $j \in \mathcal{U}$. To avoid double-counting the revenue of serving one particular cargo, we only count the revenue from a cargo j if the cargo is finally downloaded at its destination port DP_j. To represent this, we modify (48) above to

$$\sum_{s \in \mathcal{S}} \sum_{k \in \mathcal{K}_s} \sum_{j \in \mathcal{L} \cup \mathcal{U}} SR_j \; XD_{jsk}. \tag{49}$$

5.2.3 New constraints added

We also have to include some new constraints to model cargo-transfer activity. These new constraints contain bilinear terms and an exact linearization scheme. This linearization scheme used for the bilinear terms throughout this thesis can be found in the Appendix A.

At most one ship delivers a cargo to its destination port

From Figure 19, we have two possible ways to download a cargo j at its destination port DP_j: namely, when the cargo is brought directly from its pickup port PP_j, and when the cargo originates from a transfer port $i \neq PP_j$. To ensure that only one of these two events occurs, we introduce the following constraint

$$\sum_{s\in\mathcal{S}}\sum_{k\in\mathcal{K}_s} XD_{jsk}\ X_{DP_j,sk} \leq 1, \qquad j \in \mathcal{L}\cup\mathcal{U}. \tag{50}$$

One transfer port

When a cargo is served by two ships, a ship s_1 has to pickup from its pickup port PP_j and download it at a transfer port i. And later, another ship s_2 will visit that transfer port, pickup cargo j and discharge it at its destination port DP_j. Hence there are two downloads and two pickups done to cargo j. The following constraint makes sure that the first download and the second pickup happen in the same port i

$$\sum_{s\in\mathcal{S}}\sum_{k\in\mathcal{K}_s} i\ XPT_{jsk}\ X_{isk} = \sum_{s\in\mathcal{S}}\sum_{k\in\mathcal{K}_s} i\ XDT_{jsk}\ X_{isk}, \qquad j \in \mathcal{U}, i \in \mathcal{I}. \tag{51}$$

One pickup and one download

As mentioned earlier, once a ship serves a cargo, it has to pickup once and download once. Also from Figure 19, if $XP_{jsk} = 1$, then for that particular j and s, we must have $XPT_{jsk'} = 0$ for all $k' \in \mathcal{K}\backslash\{k\}$. This is reflected in the following bilinear constraint

$$XP_{jsk_1}\ XPT_{jsk_2} = 0, \qquad \text{for all } (k_1,k_2)\text{-pair}, j \in \mathcal{U}, s \in \mathcal{S}, k_1, k_2 \in \mathcal{K}_s, k_1 \neq k_2. \tag{52}$$

Similarly, a download can occur only once when a ship serves a cargo. So that

$$XD_{jsk_1}\ XDT_{jsk_2} = 0, \qquad \text{for all } (k_1,k_2)\text{-pair}, j \in \mathcal{U}, s \in \mathcal{S}, k_1, k_2 \in \mathcal{K}_s, k_1 \neq k_2. \tag{53}$$

Transfer cargo is picked-up by 2^{nd} ship after downloaded by 1^{st} ship

For each cargo transfer activity, cargo pickup by the second ship can only happen after cargo download has been done by the first ship. This is reflected in the following trilinear constraint

$$XDT_{js_1k_1}\ XPT_{js_2k_2}\ T_{s_1k_1} \leq XDT_{js_1k_1}\ XPT_{js_2k_2}\ T_{s_2k_2}, \\ j \in \mathcal{U}, s_1 \neq s_2, s_1, s_2 \in \mathcal{S}, k_1 \in \mathcal{K}_{s_1}, k_2 \in \mathcal{K}_{s_2}. \tag{54}$$

This trilinear constraint is exactly reformulated as a linear system by Proposition A.0.1 in Appendix A.

5.2.4 Price to pay

Because constraints (6) and (7) are replaced by (42) and (43), we can no longer guarantee the integrality of variables XP_{jsk} and XD_{jsk}, which were forced to integrality by (6) and (7). The same is true for variables XPT_{jsk} and XDT_{jsk}. Therefore, all of them have to be set as binary variables. The good thing is, by constraints (52) and (53), we just have to make sure either one of the pickup variables XP_{jsk} or XPT_{jsk} must be binary. The same is true for the download variables, only one of XD_{jsk} or XDT_{jsk} is set to be binary.

New binary variables, as well as all introduced constraints, will of course increase the size of the problem tremendously. By introducing the cargo transfer capability to the model, we add additional $|\mathcal{U}||\mathcal{S}||\mathcal{K}|$ integer variables and $\{|\mathcal{L}| + |\mathcal{U}| + |\mathcal{U}||\mathcal{I}| + |\mathcal{U}||\mathcal{S}||\mathcal{K}|^2(2 + |\mathcal{S}|)\}$ constraints. For our 10-ship problem, for example, the number of the constraints will increase from 127,079 to 2,685,374. And the number of integer and continuous variables increased from 3,380 and 112,560 to 9,760 and 885,780, respectively.

5.2.5 Example

We will illustrate our cargo transfer model with a small example. The problem has 3 ships and 7 cargos. The data is shown in Tables 20, 21, and 22. Time zero is August 12, 00:00AM. The key in this example is that more than one ship has one common destination that is profitable, yet quite far away. The cargo transfer capability allows ships to pool all these cargos into one ship going to that destination, while pickup of each cargo is done by a different ship. The optimal objective value without allowing cargo transfers is $969,468 while with cargo transfers, the profit increases to $1,144,308.83. The schedule chart of this 3-ship problem is depicted in Figure 20.

Table 20: Unloaded cargo information details

Cargo	Origin	Destination	Pickup time window	Volume (tonnes)	Shipping Rate (USD)	Status at time zero
1	Ulsan	Brisbane	12-14 Aug	2000	380,000	Unloaded
2	Onsan	Brisbane	12-14 Aug	2500	450,000	Unloaded
3	Taichung	Brisbane	12-15 Aug	3000	500,000	Unloaded
4	Onsan	Brisbane	-	1,000	20,000	Loaded on Ship 1
5	Onsan	Singapore	-	1,100	25,000	Loaded on Ship 1
6	Jasaan	Singapore	-	1,200	40,000	Loaded on Ship 2
7	Kaohsiung	Anyer	-	1,500	30,000	Loaded on Ship 3

This example shows that by allowing inter-ship cargo-transfers we may significantly reduce the operational costs and it also reflects more common practice.

Table 21: Ship information details for 3-ship problem

Ship	Size (dwt)	Cost (USD/day)	Immediate Destination	ETA (day)	Fuel Cost (USD/nm)	Consumption (tonnes/day at 13 knots)
1	11000	5000	Ulsan	2.525	6.18	14
2	11000	9000	Onsan	1.875	8.15	18
3	11000	11000	Taichung	1.732	7.25	16

Table 22: Port information details for 3-ship problem

Port Number	Port Name	Port Cost (USD) Ship-1	Port Cost (USD) Ship-2	Port Cost (USD) Ship-3
1	Ulsan	5,692	6,591	8,150
2	Onsan	3,688	5,587	6,262
3	Taichung	4,219	4,624	6,484
4	Singapore	5,348	7,248	12,006
5	Anyer	4,853	6,250	6,609
6	Brisbane	5,500	6,846	7,347

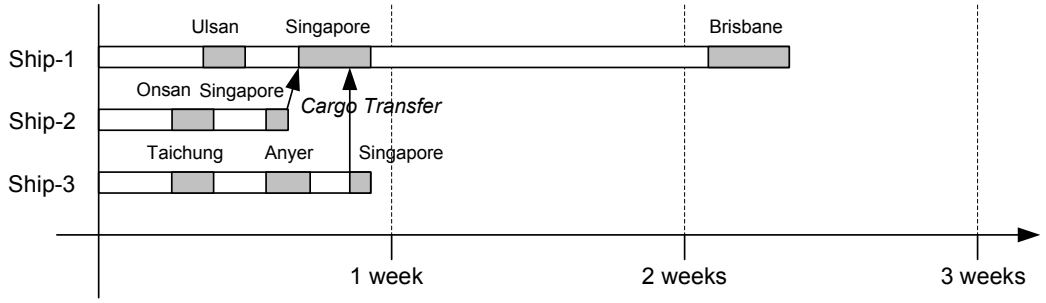

Figure 20: Gantt-charts for the 3-ship schedules with cargo transfers

CHAPTER VI

APPLICATION TO TRAIN ROUTING AND SCHEDULING PROBLEM

In this chapter we discuss the use of our model to solve a train routing and scheduling problem faced by a real railroad company. Train routing and scheduling problems involve extensive time aspects since one of the goals is to have an optimal train schedule. We extended the use of our model to answer the train routing and scheduling problem in a leading railroad company. In this application, we modify our model and add more constraints to better reflect the application. Here we also extend the model to accommodate precedence requirements. In some ways we find similarities to the problem faced by a sea-cargo company, but also due to the nature of a train's limited movements, we may reduce the number of some variables quite dramatically.

The train scheduling problem is a very large-scale network optimization problem which typically calls for trillions of decision variables. Early research on train routing and scheduling includes the work by Assad [7], Haghani [26], and Keaton [35], [36]. They divide the train scheduling problem into two separate phases. The first one is a train design problem which determines train routes, and the second is the block routing problem which determines the route blocks have to follow between the trains formed. They are solved separately in an iterative procedure with feedback from one phase to the other phase. A more recent paper by Gorman [25] suggests solving an integrated train scheduling problem heuristically.

Ahuja et.al. [1] solve a train scheduling problem by a network modeling approach. First, they assume that each train runs every day of the week. Ignoring the train timings, a network-based model is introduced. Since the number of variables and constraints are too big, they suggest a decomposition solution strategy. So they solve a

sequence of smaller decision models so that when the solutions of these smaller models are taken collectively, a good solution is obtained. The decomposition they suggest consists of two phases. In the first phase, train network and block-to-train assignments are defined. Then in phase two, train frequencies and timings are determined for the train network from phase one.

6.1 Problem description

Let us now start with the problem description of the train routing and scheduling problem of interest and how it differs from the shipping model studied until now. The first very important concept in train language is the *block* concept, which arises in the context that a railroad serves thousands or millions of shipments from their origins to respective destinations. A typical shipment consists of a set of *cars* having a common origin and destination. To reduce the handling of individual shipments as they travel, a set of shipments is grouped together as a *block*. Cars in the same block may then pass through a series of intermediate *classification yards*, being separated and reclassified only after they have reached the destination of the block. The *blocking policy*, business practices, and scheduler's experience specify what blocks should be built at each yard in the network and which cars should go into each block. The optimal *blocking plan* and assignment of cars to trains are beyond the scope of this research.

When a train passes through an intermediate classification yard, it may download or pickup blocks of cars. A block left by an inbound train is either transferred or *swapped* to a different train or it is broken up and its cars are reclassified. Some cars may have reached their destination and some are being transferred to another block. Note here that although the origin and destination of a block may correspond to those of a train, a block may also switch trains several times before reaching its final destination. We call each block switch from one train to another a *swap*. Usually, the railroad company has a business practice limiting the maximum number of swaps a

block can have during its journey.

In our train routing and scheduling problem, we are given a fleet of trains $\mathcal{S}$ with different characteristics. Without loss of generality, we may assume that the fleet has homogeneous characteristics. At time zero, we know the current locations of all trains. A train can only originate from certain stations and terminate at certain stations.

Trains have to carry blocks from their origins to specified destinations. Let $\mathcal{B}$ be the set of blocks, each block has different characteristics (origin, destination, tonnage/weight, height, width, daily volume). A train $s \in \mathcal{S}$ can visit a collection of stations from the set $\mathcal{I}$. When a train visits a station $i \in \mathcal{I}$, there are several activities done to the train, namely pickup, download/setoff, crew change, and most likely some inspections. Typically, once a train arrives at a station, it waits for an inspection to be done by station officers. After the train passes the inspection procedure, some blocks can be detached from the train, then it may proceed to pickup some other blocks. Before the train may leave the station to continue its voyage, it must again go through some inspections. We denote the total inspection time and crew-change of any train in any station as T_{adm}. If there is a pickup, the train has to spend another t_{pickup}, which does not depend on the number of cars or blocks the train picks up. Similarly, $t_{download}$ is the time spent for a download/set-off operation.

The objective is to assign all block-segments to be served by available trains and determine the route that each train should follow during the planning horizon, such that the number of block-swaps (block transfer between trains) is minimized. There are also some other objectives to be minimized; i.e., number of trains running, the length of trains operating schedules, cost of block-to-train assignments.

To use our model to solve this train routing and scheduling problem, we break down a block into several block-segments and define precedence relationships among them. Each block-segment has its own origin and destination. The reader is advised to differentiate between the origin-destination pair of a block-segment and that of the

block itself. We will refer to the latter as a block-origin and block-destination pair to avoid confusion.

Based on its block-path, each block $b \in \mathcal{B}$ can be broken down into a sequence of $|b|$ block-segments, namely $j_1 \rightarrow j_2 \rightarrow ... \rightarrow j_{(|b|-1)} \rightarrow j_{|b|}$. And let $\mathcal{J}_b$ be the set $\{j_1, j_2, ..., j_{|b|}\}$. Each block-segment $j \in \mathcal{U}$ has its pickup station and destination station information, denoted by PP_j and DP_j, respectively.

The meaning of the term *leg* is the same as the one used to describe the ship routing and scheduling problem. The number of stations visited (equal to the number of legs) is also predetermined by the schedule planner. The output of this model will be used later to define the train frequencies. We follow the formulation (1)-(14) of our original model and make some changes to the rest of the constraints.

6.2 Model formulation

A schedule for train s is presented as a series of $(K_s + 1)$ legs, where K_s is the number of legs a planner plans to schedule, or we may say that this is the maximum number of stations a train could visit in a journey. We denote $k = 0$ as the leg-0, representing the leg where a train reaching its first station at time zero. If the train is already at some originating station, the first station would be the originating station itself. This initial target station information is given prior to the new schedule generation. Let $\mathcal{K}_s = \{0, 1, ..., K_s\}$ be the set of legs for train s.

In addition to real stations, a dummy station $i = 0$ is introduced with zero station cost and zero distance from any other station. There is no block originating from and to be discharged at this dummy station. Therefore the model also forces a condition that after a train visits this dummy station, it will stay there until the end of the planning horizon. We recognize this dummy station as an absorbing station used to idle any train, if this option is considered profitable.

One interesting feature of this formulation, which makes the model different from the traditional network flow approach, is that time is treated as continuous variables

as opposed to discrete units. This makes it possible to get train routes and schedules simultaneously in one model. The model is a nonlinear model, but an *exact* linearization scheme will be introduced. And we also introduce the decision variables along with the explanation of the constraints.

6.2.1 Routing constraints

The routing constraints define and link the sequence of arrivals and departures of the various trains to and from available stations. We define binary variables X_{isk_s} to model whether or not train s is at station i at the end of sailing leg k_s. Note that the index k depends on s, because for different trains we may have different numbers of legs ($k_s \in \mathcal{K}_s$), where $\mathcal{K}_s$ denotes the set of legs for train s. But for simplicity, let us drop the subscript s from index k_s. We define

$$X_{isk} = \begin{cases} 1 & \text{if train } s \text{ reaches station } i \text{ at the end of leg } k \\ 0 & \text{otherwise} \end{cases}$$

where T_{sk} denotes a continuous variable indicating the time at which train s arrives at any station at the end of leg k.

Visit exactly one station in each leg

A train can only visit one station during each leg. It can be any of the service stations or the dummy station

$$\sum_i X_{isk} = 1, \qquad s \in \mathcal{S}, k \in \mathcal{K}_s. \tag{55}$$

Maximum one visit for all stations $i \in \mathcal{I} \setminus \{0\}$

We may relax this assumption later to accommodate multiple visits to one station. Now, assume that a train can visit a specific station at most once during each leg, except for the dummy station, so

$$\sum_{k} X_{isk} \leq 1, \qquad i \in \mathcal{I} \setminus \{0\}, s \in \mathcal{S}. \tag{56}$$

Dummy station as an absorbing station

To prevent a train from transiting through the dummy station, we force that once a train visits the dummy station, it has to stay there until the end of the planning horizon. In other words, the dummy station is only visited by trains that are idle until the end of planning horizon. We express this in the following constraint:

$$X_{0sk} \leq X_{0s(k+1)}, \qquad s \in \mathcal{S}, k \in \mathcal{K}_s \backslash \{K_s\}. \tag{57}$$

Congestion issue in a station

For the planning horizon, the number of trains visiting station $i \in \mathcal{I}$ cannot be more than certain limit, CS_i. That is,

$$\sum_{s} \sum_{k} X_{isk} \leq CS_i, \qquad i \in \mathcal{I} \setminus \{0\}. \tag{58}$$

Station-to-station travel

To model travel from the current station to the next station, binary transition variables are defined as

$$Z_{ilsk} = \begin{cases} 1 & \text{if train } s \text{ visits station } i \text{ at the end of leg } k \\ & \text{and station } l \text{ at the end of leg } (k+1) \\ 0 & \text{otherwise.} \end{cases}$$

Clearly, $Z_{ilsk} = X_{isk}X_{ls(k+1)}$, for all $i, j \in \mathcal{I}, s \in \mathcal{S}, k \in \mathcal{K}_s \backslash \{K_s\}$. This represents a nonlinear relationship, but we utilize an *equivalent* linear reformulation of this type of constraint given by these two constraints:

$$\sum_{l \in \mathcal{I}, l \neq i} Z_{ilsk} = X_{isk}, \qquad i \in \mathcal{I}, s \in \mathcal{S}, k \in \mathcal{K}_s \backslash \{K_s\}, \tag{59}$$

$$\sum_{i \in \mathcal{I}, i \neq l} Z_{ilsk_s} = X_{ls(k_s+1)}, \qquad l \in \mathcal{I}, s \in \mathcal{S}, k \in \mathcal{K}_s \backslash \{K_s\}. \tag{60}$$

By this equivalent linear formulation, we may treat Z_{ilsk} as a $0-1$ *continuous variable.*

Line capacity constraint

Between two stations i and l, there are limited number C_{il} of trains passing through, that is,

$$\sum_{s} \sum_{k} Z_{ilsk} = CL_{il}, \qquad i, l \in \mathcal{I}. \tag{61}$$

Maximum detour constraint

Suppose $|S_{path}|$ denotes the length of the shortest path from the origin to the destination of block b. But the actual route the block will follow after it is assigned to some trains might not follow the shortest path. We say here that the block route is being snapped to the train route. Some business rules might impose the detour constraint of α miles. It means that the route that a block actually follows must be less than or equal to α plus the distance of the original shortest path route. That is,

$$\sum_{b \in \mathcal{J}_b} \sum_{s} \sum_{k} XC_{jsk}\ TT_{sk}\ v_s \leq |S_{path}| + \alpha, \qquad b \in \mathcal{B}. \tag{62}$$

and recall that v_s denotes the velocity of the train s. Here we need a new variable as a relaxation to the bilinear relationship of variables XC_{jsk} and TT_{sk}. Let $XCTT_{jsk}$ be such variable. Then for each block $b \in \mathcal{B}$, the bilinear term $XC_{jsk}\ TT_{sk}$ in (62)

can be replaced by $XCTT_{jsk}$ and the constraints augmented by the system of linear inequalities (see Appendix A):

$$\begin{aligned} XCTT_{jsk} &\geq 0 \\ XCTT_{jsk} &\geq TT_{sk} + H * XC_{jsk} - H \\ XCTT_{jsk} &\leq TT_{sk} \\ XCTT_{jsk} &\leq H * XC_{jsk} \end{aligned}$$

where H denotes the time horizon of the planning period. Also note that this system involves more constraints but *no* new integer variables.

A scheduler might also want to differentiate the velocity of a train between one segment from another. A constraint similar to (62) is

$$\sum_{b\in\mathcal{J}_b}\sum_{s}\sum_{k}\sum_{i}\sum_{j}\frac{Dis_{il}}{24v_{ijs}}\ XC_{jsk}\ Z_{ijsk} \leq |S_{path}| + \alpha, \qquad b \in \mathcal{B}. \tag{63}$$

Here, Dis_{il} and v_{ils} denote the distance from stations i to l and the velocity of the train s on the line segment connecting stations i to l, respectively. Let $XCZ_{jilsk} = XC_{jsk}\ Z_{ilsk}$, then a system of linear relaxations of this constraint for every combination of j, i, l, s, k is

$$\begin{aligned} XCZ_{jsk} &\geq 0 \\ XCZ_{jsk} &\geq XC_{jsk} + Z_{ilsk} - 1 \\ XCZ_{jsk} &\leq XC_{jsk} \\ XCZ_{jsk} &\leq Z_{ilsk} \end{aligned}$$

Again, *no* new integer variables are involved, but adding the relaxation system to the model is surely pricey.

6.2.2 Block-segment movement constraints

We define binary variables Y_{js} to indicate whether a train s should serve a block-segment j:

$$Y_{js} = \begin{cases} 1 & \text{if train } s \text{ serves block-segment } j \\ 0 & \text{otherwise.} \end{cases}$$

In our current scenario, once a train picks up a block-segment from its pickup station, it has to carry it all the way to the destination station within the time horizon. To capture these movements, we define three types of binary variables XP_{jsk}, XD_{jsk}, and XC_{jsk} that later turn out can be set as $0-1$ continuous variable.

$$XP_{jsk} = \begin{cases} 1 & \text{if train } s \text{ picks up block-segment } j \text{ at the end of leg } k \\ 0 & \text{otherwise} \end{cases}$$

$$XD_{jsk} = \begin{cases} 1 & \text{if train } s \text{ downloads block-segment } j \text{ at the end of leg } k \\ 0 & \text{otherwise} \end{cases}$$

$$XC_{jsk} = \begin{cases} 1 & \text{if train } s \text{ carries block-segment } j \text{ onboard during leg } k \\ 0 & \text{otherwise} \end{cases}$$

Block-segment pickups and downloads

For a train s to load a block-segment j at the end of leg k, the train must visit the loading station PP_j at time T_{sk}, and of course it must service block-segment j. Hence, we must have

$$XP_{jsk} = Y_j X_{(PP_j)sk}, \qquad j \in \mathcal{U}, s \in \mathcal{S}, k \in \mathcal{K}_s.$$

Similarly, for a train s to unload a block-segment j at the end of leg k, the train must visit the unloading station DP_j at time T_{sk}, and it must service block-segment j. Therefore,

$$XD_{jsk} = Y_{js} X_{(DP_j)sk}, \qquad j \in \mathcal{U}, s \in \mathcal{S}, k \in \mathcal{K}_s.$$

These non-linear constraints are replaced by the following linear constraints. Firstly, when a train s does service a block-segment j (i.e., $Y_{js} = 1$), then it must be loaded in exactly one leg. Otherwise, when it does not serve the block-segment, then it cannot be loaded in any leg. The same is true for the download. Thus,

$$\sum_k XP_{jsk} = Y_{js}, \qquad j \in \mathcal{U}, s \in \mathcal{S}, \tag{64}$$

$$\sum_k XD_{jsk} = Y_{js}, \qquad j \in \mathcal{U}, s \in \mathcal{S}. \tag{65}$$

Secondly, the pickup and download can happen only at a block-segment's pickup and discharge stations, respectively. That is,

$$XP_{jsk} \leq X_{(PP_j)sk}, \qquad j \in \mathcal{U}, s \in \mathcal{S}, k \in \mathcal{K}_s, \tag{66}$$

$$XD_{jsk} \leq X_{(DP_j)sk}, \qquad j \in \mathcal{U}, s \in \mathcal{S}, k \in \mathcal{K}_s. \tag{67}$$

By these constraints we are assured that we can treat variables XP_{jsk} and XD_{jsk} as 0-1 *continuous variables*.

Pickup before download

If a train services a block-segment, then it must visit the pickup station before it visits the delivery station, which is expressed in the following constraint:

$$\sum_{k>0} k(XD_{jsk} - XP_{jsk}) \geq Y_{js}, \qquad j \in \mathcal{U}, s \in \mathcal{S} \tag{68}$$

Carrying block-segment from pickup to download station

Also, to make a train carry a block-segment j from its pickup station to its delivery station, the following constraint is used:

$$XC_{js(k+1)} = XC_{jsk} + XP_{jsk} - XD_{jsk}, \qquad j \in \mathcal{L} \cup \mathcal{U}, s \in \mathcal{S}, k \in \mathcal{K}_s \backslash \{K_s\} \tag{69}$$

Since variables XP_{jsk} and XD_{jsk} are automatically set to be binary, the same applies to variable XC_{jsk}.

Train capacity

Every time a train carries some block-segments onboard, they must be within the carrying capacity of the train. Thus,

$$\sum_j V_j XC_{jsk} \leq VMAX_s, \qquad s \in \mathcal{S}, k \in \mathcal{K}_s. \tag{70}$$

One block-segment is served by one train

To ensure that a block-segment is serviced by at most one train, the following constraint is imposed:

$$\sum_s Y_{js} = 1, \qquad j \in \mathcal{U} \tag{71}$$

6.2.3 Time constraints

We let T_{sk} denote the time at which a train s arrives at any station at the end of its leg k. After a train performs its service at a station, it will continue its journey to the next station. Let TT_{sk} denote the time a train s takes to travel from a current station at the end of leg k to the next station at the end of leg $(k+1)$. If the distance between station i and station l is Dis_{il}, then TT_{sk} can be expressed as

$$TT_{sk} = \sum_i \sum_l \frac{\text{Dis}_{il} Z_{ilsk}}{24 v_{ijs}} \qquad s \in \mathcal{S}, k \in \mathcal{K}_s \backslash \{K_s\} \tag{72}$$

Here, v_{ijs} denotes the speed of train s on the line segment connecting stations i and j. Notice that only one Z_{ilsk} can equal 1 for all i and all j. Hence, TT_{sk} is actually the distance between two stations visited at the ends of leg k and $(k+1)$ divided by the average hourly speed of the train.

Time spent at a station

The time spent at a station really depends on the operations needed to be performed within the station. The total time spent in a station must exceed the time required for inspections, plus the time for discharging all delivery block-segments, plus the time for loading pickup block-segments. Pickup and download activities consume t_{pickup} and $t_{download}$ units of time, respectively. And particularly for a dummy station, we do not have administrative time. Hence,

$$T_{s(k+1)} \geq T_{sk} + TT_{sk} + T_{adm}(1 - X_{0sk}) + t_{pickup}\ P_{sk} + t_{download}\ D_{sk},$$
$$s \in \mathcal{S}, k \in \mathcal{K}_s \backslash \{K_s\}, \qquad (73)$$

where T_{adm} indicates administrative/inspection time. The new variables P_{sk} and D_{sk} are indicator variables telling us whether a pickup and a download, respectively, happen to train s at the end of leg k:

$$P_{sk} = \begin{cases} 1 & \text{if train } s \text{ picks up some blocks at the end of leg } k \\ 0 & \text{otherwise} \end{cases}$$

$$D_{sk} = \begin{cases} 1 & \text{if train } s \text{ picks up some blocks at the end of leg } k \\ 0 & \text{otherwise} \end{cases}$$

With some analysis, for every block $b \in \mathcal{B}$, and for every consecutive pair (j_1, j_2) such that $j_1, j_2 \in \mathcal{J}_b$, we may write P_{sk} and D_{sk} as follows:

$$P_{sk} = max\left\{max_{j_1 \rightarrow j_2, j_1, j_2 \in \mathcal{J}_b, b \in \mathcal{B}}\{XC_{j_2 s(k+1)} - XC_{j_1 sk}\}, max_{j \in \mathcal{F}} XP_{jsk}\right\},$$

$$D_{sk} = max\left\{max_{j_1 \to j_2, j_1, j_2 \in \mathcal{J}_b, b \in \mathcal{B}}\{XC_{j_1 sk} - XC_{j_2 s(k+1)}\}, max_{j \in \mathcal{L}} XD_{jsk}\right\},$$

where $j_1 \to j_2$ indicates that block-segment j_1 is the predecessor of block-segment j_2. The sets $\mathcal{F}$ and $\mathcal{L}$ denote the set of first and last block-segments in all block $b \in \mathcal{B}$, respectively. For all $s \in \mathcal{S}, k \in \mathcal{K}_s$, an equivalent linear system of P_{sk} is:

$$\begin{aligned} P_{sk} &\geq XC_{j_2 s(k+1)} - XC_{j_1 sk}, \quad j_1, j_2 \in \mathcal{J}_b, j_1 \to j_2, b \in \mathcal{B}, \\ P_{sk} &\geq XP_{jsk}, \qquad\qquad\qquad\quad j \in \mathcal{F}. \end{aligned}$$

Similarly, D_{sk} can be represented by equivalent linear system:

$$\begin{aligned} D_{sk} &\geq XC_{j_1 sk} - XC_{j_2 s(k+1)}, \quad j_1, j_2 \in \mathcal{J}_b, j_1 \to j_2, b \in \mathcal{B}, \\ D_{sk} &\geq XD_{jsk}, \qquad\qquad\qquad\quad j \in \mathcal{L}. \end{aligned}$$

for all $s \in \mathcal{S}, k \in \mathcal{K}_s$.

Number of block-swaps for each block

A block-swap is defined if there is a change in trains carrying a block. For example, if a block b is carried by train t_1 from station A to station B and continues from station B to station C by train t_2, we say that there is a block-swap for block b from train t_1 to train t_2 at station B. Define the following variable to capture the number of non-swaps between block-segments so we may use this to calculate the number of block-swaps in the block,

$$W_{j_1 j_2 s} = \begin{cases} 1 & \text{if block-segment } j_1 \text{ is an immediate predecessor of block-segment } j_2 \\ & \text{and train } s \text{ serves both} \\ 0 & \text{otherwise} \end{cases}$$

and the number of block-swaps SWAP_b for block b, satisfies the following equation

$$\text{SWAP}_b = |b| - \sum_{s} \sum_{j_1, j_2 \in \mathcal{J}_b, j_1 \to j_2} W_{j_1 j_2 s} - 1. \tag{74}$$

Clearly, $W_{j_1 j_2 s} = Y_{j_1 s} Y_{j_2 s}$ and the following equivalent linear system will be used:

$$\begin{aligned} W_{j_1 j_2 s} &\geq 0 \\ W_{j_1 j_2 s} &\geq Y_{j_1 s} + Y_{j_2 s} - 1 \\ W_{j_1 j_2 s} &\leq Y_{j_1 s} \\ W_{j_1 j_2 s} &\leq Y_{j_2 s} \end{aligned}$$

for every $j_1, j_2 \in \mathcal{J}_b$, such that $j_1 \rightarrow j_2$, for all $s \in \mathcal{S}$ and $k \in \mathcal{K}_s$.

The company business rules often define a limit on the number of block-swaps which are used as a measure of service level and can affect customer satisfaction. If we limit the number of block-swaps to be Max_{swap}, we may impose this constraint to the model as

$$\text{SWAP}_b \leq \text{Max}_{swap}, \ b \in \mathcal{B}. \tag{75}$$

Block-Swap Timing

For every consecutive block-segment pair $(j_1, j_2) \in \mathcal{J}_b, b \in \mathcal{B}$, from the train point of view, the set-off of the earlier block-segment j_1 has to happen before the pickup of the later block-segment j_2. Moreover, a block being swapped spends several hours of connection handling time. In other word, once a block is set-off by some train, it cannot be picked-up by another train in the next δ_{swap} hours, for example. Hence, we have that for each pair (j_1, j_2) such that $j_1 \rightarrow j_2$,

$$\sum_{s \in \mathcal{S}} \sum_{k \in \mathcal{K}_s} T_{sk} XD_{j_1 sk} + \delta_{swap} \leq \sum_{s \in \mathcal{S}} \sum_{k \in \mathcal{K}_s} T_{s(k+1)} XP_{j_2 s(k+1)}. \tag{76}$$

Again, this is a nonlinear constraint. For every $s \in \mathcal{S}$ and $k \in \mathcal{K}_s$, let $TD_{j_1 sk} = T_{sk} XD_{j_1 sk}$ and $TP_{j_2 sk} = T_{sk} XP_{j_2 sk}$ be the predecessor's set-off time and the successor's pickup time, respectively. And let H be the planning horizon. Then an equivalent linear system of constraint (76) for each (j_1, j_2) pair is

$$
\begin{aligned}
\sum_{s\in\mathcal{S}}\sum_{k\in\mathcal{K}_s} TD_{j_1sk} + \delta_{swap} &\leq \sum_{s\in\mathcal{S}}\sum_{k\in\mathcal{K}_s} TP_{j_2s(k+1)} \\
TP_{j_2s(k+1)} &\geq 0 \\
TP_{j_2s(k+1)} &\geq T_{s(k+1)} + H * XP_{j_2s(k+1)} - H \\
TP_{j_2s(k+1)} &\leq H * XP_{j_2s(k+1)} \\
TP_{j_2s(k+1)} &\leq T_{s(k+1)} \\
TD_{j_1sk} &\geq 0 \\
TD_{j_1sk} &\geq T_{sk} + H * XD_{j_1sk} - H \\
TD_{j_1sk} &\leq H * XD_{j_1sk} \\
TD_{j_1sk} &\leq T_{sk}
\end{aligned}
$$

6.2.4 Objective function

The objective is to minimize the costs of the weighted sum of the following components: number of block-swaps, number of trains running, the length of train's operating schedules, and cost of block-to-train assignment. The weight given is naturally given by the dollar value of the corresponding costs.

Total number of block-swaps

Denoting the cost of performing a swap by TS_b, we may capture the cost of block-swaps as

$$\sum_{b\in\mathcal{B}} TS_b \text{ SWAP}_b. \tag{77}$$

Total cost of train starts

This captures the one time cost of running a train. It might relate to the setup cost, crew cost, etc. Let TC_s be the one-time cost of running train s. Since an operating

train does not visit the dummy station at the end of leg-1, we may calculate this cost as

$$\sum_{s} TC_s(1 - X_{0s1}). \tag{78}$$

Total cost of train's trip hours

We may also have the variable cost of running a train per unit time (TR_s); for example, the rental fee for a locomotive. We also want this cost to be minimized. We capture this cost by

$$\sum_{s} TR_s \; T_{sK_s}. \tag{79}$$

Total cost of train miles

We may also want to include the cost incurred when the train is moving from one station to another. An example of this is the fuel cost per mile. This cost depends on the travel time or distance between stations. Let TF_{sk} be the travel cost incurred by train s during leg k, then we may add this term to the objective function

$$\sum_{s \in \mathcal{S}} \sum_{k \in \mathcal{K}_s} TF_{sk} \; TT_{sk}. \tag{80}$$

Cost of block-to-train assignments

There might also be some costs related to the block-to-train assignments, meaning that the cost of moving one particular block-segment by one train differs from that of another train. We may express this as

$$\sum_{b\in\mathcal{B}}\sum_{j\in\mathcal{J}_b}\sum_{s\in\mathcal{S}} TA_{js}\ Y_{js}, \tag{81}$$

where TA_{js} denotes the cost of assigning block-segment j to train s. This completes the formulation for the train routing and scheduling problem as an MILP problem. One good thing about this train routing and scheduling problem is that trains can only run on specific tracks, hence we can reduce the number of variables Z_{ilsk} tremendously.

6.2.5 Complete train routing and scheduling formulation

The complete MINLP formulation for train routing and scheduling is given as follow.

$$(OBJ)\ \text{Minimize}\ \begin{cases} \sum_{b\in\mathcal{B}} TS_b\ \text{SWAP}_b & \text{Number of block-swaps} \\ +\sum_s TC_s(1-X_{0s1}) & \text{Cost of train starts} \\ +\sum_s TR_s\ T_{sK_s} & \text{Cost of train's trip hours} \\ +\sum_{s\in\mathcal{S}}\sum_{k\in\mathcal{K}_s} TF_{sk}\ TT_{sk} & \text{Cost of train miles} \\ +\sum_{b\in\mathcal{B}}\sum_{j\in\mathcal{J}_b}\sum_{s\in\mathcal{S}} TA_{js}\ Y_{js} & \text{Cost of block-to-train assignments} \end{cases}$$

$$\sum_i X_{isk} = 1, \qquad s\in\mathcal{S}, k\in\mathcal{K}_s$$

$$\sum_k X_{isk} \le 1, \qquad i\in\mathcal{I}\setminus\{0\}, s\in\mathcal{S}$$

$$X_{0sk} \le X_{0s(k+1)}, \qquad s\in\mathcal{S}, k\in\mathcal{K}_s\setminus\{K_s\}$$

$$\sum_s\sum_k X_{isk} \le CS_i, \qquad i\in\mathcal{I}\setminus\{0\}$$

$$\sum_{l\in\mathcal{I},l\neq i} Z_{ilsk} = X_{isk}, \qquad i\in\mathcal{I}, s\in\mathcal{S}, k\in\mathcal{K}_s\setminus\{K_s\}$$

$$\sum_{i\in\mathcal{I},i\neq l} Z_{ilsk_s} = X_{ls(k_s+1)}, \qquad l\in\mathcal{I}, s\in\mathcal{S}, k\in\mathcal{K}_s\setminus\{K_s\}$$

$$\sum_s\sum_k Z_{ilsk} = CL_{il}, \qquad i,l\in\mathcal{I}$$

$$\sum_{b\in\mathcal{J}_b}\sum_s\sum_k XC_{jsk}\ TT_{sk}\ v_s \le |S_{path}| + \alpha, \qquad b\in\mathcal{B}$$

$$\sum_{b\in\mathcal{J}_b}\sum_s\sum_k\sum_i\sum_j \frac{Dis_{il}}{24v_{ijs}}\ XC_{jsk}\ Z_{ijsk} \le |S_{path}| + \alpha, \qquad b\in\mathcal{B}$$

$$XP_{jsk} = Y_j X_{(PP_j)sk}, \qquad j\in\mathcal{U}, s\in\mathcal{S}, k\in\mathcal{K}_s$$

$$XD_{jsk} = Y_{js} X_{(DP_j)sk}, \qquad j\in\mathcal{U}, s\in\mathcal{S}, k\in\mathcal{K}_s$$

$$\sum_k XP_{jsk} = Y_{js}, \qquad j \in \mathcal{U}, s \in \mathcal{S}$$

$$\sum_k XD_{jsk} = Y_{js}, \qquad j \in \mathcal{U}, s \in \mathcal{S}$$

$$XP_{jsk} \leq X_{(PP_j)sk}, \qquad j \in \mathcal{U}, s \in \mathcal{S}, k \in \mathcal{K}_s$$

$$XD_{jsk} \leq X_{(DP_j)sk}, \qquad j \in \mathcal{U}, s \in \mathcal{S}, k \in \mathcal{K}_s$$

$$\sum_{k>0} k(XD_{jsk} - XP_{jsk}) \geq Y_{js}, \qquad j \in \mathcal{U}, s \in \mathcal{S}$$

$$XC_{js(k+1)} = XC_{jsk} + XP_{jsk} - XD_{jsk}, \qquad j \in \mathcal{L} \cup \mathcal{U}, s \in \mathcal{S}, k \in \mathcal{K}_s \backslash \{K_s\}$$

$$\sum_j V_j XC_{jsk} \leq VMAX_s, \qquad s \in \mathcal{S}, k \in \mathcal{K}_s$$

$$\sum_s Y_{js} = 1, \qquad j \in \mathcal{U}$$

$$TT_{sk} = \sum_i \sum_l \frac{\text{Dis}_{il} Z_{ilsk}}{24 v_{ijs}} \qquad s \in \mathcal{S}, k \in \mathcal{K}_s \backslash \{K_s\}$$

$$T_{s(k+1)} \geq T_{sk} + TT_{sk} + T_{adm}(1 - X_{0sk}) + t_{pickup}\ P_{sk} + t_{download}\ D_{sk},$$

$$s \in \mathcal{S}, k \in \mathcal{K}_s \backslash \{K_s\}$$

$$P_{sk} = max\left\{max_{j_1 \to j_2, j_1, j_2 \in \mathcal{J}_b, b \in \mathcal{B}}\{XC_{j_2 s(k+1)} - XC_{j_1 sk}\}, max_{j \in \mathcal{F}} XP_{jsk}\right\},$$

$$D_{sk} = max\left\{max_{j_1 \to j_2, j_1, j_2 \in \mathcal{J}_b, b \in \mathcal{B}}\{XC_{j_1 sk} - XC_{j_2 s(k+1)}\}, max_{j \in \mathcal{L}} XD_{jsk}\right\},$$

$$\text{SWAP}_b = |b| - \sum_s \sum_{j1, j_2 \in \mathcal{J}_b, j_1 \to j_2} W_{j_1 j_2 s} - 1$$

$$\text{SWAP}_b \leq Max_{swap},\ b \in \mathcal{B}$$

$$\sum_{s \in \mathcal{S}} \sum_{k \in \mathcal{K}_s} T_{sk} XD_{j_1 sk} + \delta_{swap} \leq \sum_{s \in \mathcal{S}} \sum_{k \in \mathcal{K}_s} T_{s(k+1)} XP_{j_2 s(k+1)}$$

$$\sum_{s \in \mathcal{S}} \sum_{k \in \mathcal{K}_s} T_{sk} XD_{j_1 sk} + \delta_{switch} \leq \sum_{s \in \mathcal{S}} \sum_{k \in \mathcal{K}_s} T_{sk} XP_{j_2 sk}$$

6.2.6 Example and heuristic method

We tested our model to answer a real train routing and scheduling problem faced by a U.S. railroad company. The company currently has an optimization based decision support system to find the optimal block route for all blocks. Given the optimal block routes and all other block information, the task is to create trains, along with their schedules, that serve all blocks such that all constraints are satisfied. Note here

that the final route that a block follows may not be the initial block route because a block actually travels on a train and has to follow the train route. The company prefers to have trains operating under a certain regular daily schedule. Hence it is natural to consider a daily train problem. The outputs are optimal daily schedules of trains. By repeating this train daily schedules every day of the week we may obtain a weekly train schedule. However, we may want to eliminate some small-volume trains to reduce operating expenses. Here we may again use our model to decide how the blocks from the eliminated trains will be rerouted using the other available trains, or confirm that the low-volume trains are actually needed because eliminating them will make the problem infeasible.

We tested our model to solve the company's 63-block problem, involving 123 block-segments and 60 precedence constraints. Since it takes too much time to solve the problem to optimality, we introduce a heuristic approach. Ideally, the problem is solved simultaneously to find the optimal solution. But since this deals with a large number of integer variables, the problem becomes harder to solve. Another issue that increases the number of integer variables is the number of trains that we create as an input to the model. Trains may originate from certain stations and each originating station has its maximum limit of train starts per day. Ideally, we setup all possible trains and let the model decide which to use. However, the complexity of the problem arises when we deal with a large number of trains, where most of them might be left unused.

In our heuristic, we create a block priority list and only solve a couple of blocks in one iteration. Also, trains are only created as needed. In this approach, we gradually assign more and more blocks to trains and create more and more trains as the number of iteration increases. By this approach, as the number iterations increases, the problem becomes harder but we have more and more fixed values of variables carried out from previous iterations.

The company may have some business rules that can be used as a guide to create

the block priority list. However, in this implementation, we use a simple priority rule that is the number of block-segments within a block. The motivation is that the earlier a block is assigned permanently to some trains, the less possibility it will experience block-swaps. Thus, our list lets blocks with a small number of block-segments be processed earlier and those with a large number of block-segments later.

Our network involves 398 stations and 1933 arcs. But since trains can only move on specific tracks, our network is not a complete network as we found earlier in our sea-cargo problem. Hence the movement variables can be reduced tremendously. We have 40 potential swap locations, 85 train origins and 84 train terminals. Each train originating station may have at most 6 train starts per day. The block-swap and block-switch connecting times are 4 and 8 hours, respectively. The number of maximum legs per train is initially set to be between 12 and 15. The problem is solved by CPLEX 9.1 using one processor on a four-CPU Sun E450 server machine under all of the default options of the CPLEX.

We let $3 - 5$ blocks solve simultaneously. In each new iteration, we fix block-to-train assignments from previous iterations and create more trains as needed. A train is typically created from an originating station when another train from the originating station was used by the model in previous iteration. At the last iteration, we have a total of 53 trains created (but not all of them used). It takes $13,476$ seconds (3.75 hours) for the heuristic to solve our problem. We do hope to get a faster solution time to find the solution. But since this is a zero-based train scheduling approach - meaning that the routes and schedules are built from scratch - and the company might only perform this calculation at most 2 times a year, the computational time of 3.75 hours may still be acceptable. In real situations, the company has set some trains that must be run, based on experience, so we have a lot of variables set in the beginning which would make our heuristics run faster.

Our result shows a total number of 23 train starts, 27 block-swaps, and 22 total blocks being swapped. The number of legs used by each train ranges from 2 to 15.

Table 23: Trains information for 63-block problem

Train Number	Minimum # cars in trip plan	Maximum # cars in trip plan	Total Legs
1	108	122	7
2	30	86	7
3	14	65	7
4	112	112	3
5	49	80	6
6	11	128	13
7	4	55	13
8	44	87	9
9	18	18	12
10	4	50	6
11	2	25	12
12	2	9	12
13	39	47	9
14	7	83	12
15	11	11	6
16	5	5	2
17	8	15	12
18	14	14	12
19	7	11	12
20	10	52	12
21	6	6	12
22	20	20	8
23	5	5	15

Total train hours are from 11.25 to 56.06 hours, and the total train miles are between 225 and 812 miles. Times at which a train arrives and departs from each stations in each of its legs are also obtained as part of the output of our model. Table 23 shows us the results of the model. We may also observe the minimum and maximum number of cars loaded during the trip plan for each train. Note that a train may carry a different number of cars from one leg to another, depending on the block being attached or detached in each leg. We also include the number of legs actually used by each train.

Among those 23 trains, we can further improve the solution found so far by eliminating the low-volume trains. A company business rule defines a low-volume train as one that carries less than 40 cars in its trip plan. Applying this rule to the result displayed in Table 23, we find 10 low-volume trains, namely train 9, 11-12, 15-19, 21-23. The existence of these trains are questionable since we may transfer the loads

carried by these trains to the other available trains so that the number of train starts may be reduced. To justify, we will again use our model to see if the loads being carried by low-volume trains can actually be transferred to other trains. So now, we already have some must-have trains in our system and some blocks permanently assigned to the must-have trains. Our model will try to assign unassigned blocks to the must-have trains or decide that low-volume trains are indeed needed. First, let us discuss a few possible reasons why a low-volume train is formed:

(i) The train is actually a must-have train. For example, no other trains pass through the pickup station and it is not reasonable to reroute any other trains to visit the station.

(ii) Some trains do not have enough leg to finish its operations, therefore extra trains are needed. We know that the number of maximum legs a train can have is predetermined as an input to the model. By adding some more legs to other trains, we may eliminate this kind of low-volume train.

(iii) In some of our heuristic iterations, there is not enough trains available from specific originating station. For example, all trains originating from some specific station A may have already been used to serve other blocks, but the system actually demands another train originating from station A. Thus, the system decides to use a train from neighboring station B as a substitute. In the real situation, we may easily setup another train originating from station A to fulfill the demand. In our solution, a train may travels empty from station B, or from some other station if there are no more trains available from the near-by locations.

(iv) Because of the heuristics, at an earlier time it is considered good to assign one block to this train, but at a later point in time, serving the block by another train is better.

There are no easy ways to find out which of the above mentioned possible causes is the reason for a low-volume train. To identify the cause, we have to take a closer look to the solution in every iteration of our algorithm. Having looked closer to the solution in each iteration of our 63-block problem, we revise some input values (e.g., add more legs to some trains, make more trains available from certain locations) and rerun the model. We observe that it is actually possible to eliminate 4 low-volume trains, leaving us 7 must-have low-volume trains.

6.2.7 More computational results

We apply the foregoing heuristic on several test problem to check its performance. We randomly generated the problems consisting of 5, 10, 15, and 20 blocks. We let the problems run to optimality and the quality of the heuristic method is measured in terms of percentage from the optimal solution. We include two kinds of measurements which are solution quality and solution time. The solution quality generated by the proposed heuristic is measured as

$$\frac{\text{Heuristic solution}}{\text{Optimal solution}} \times 100\%.$$

Since our problem is a minimization problem, the measurement of solution quality would have values greater than or equal to 100%. The closer a value to 100%, the better it is. Similarly, solution time of the heuristic is measured as a comparison to the time needed to solve the problem to optimality, that is

$$\frac{\text{Time to solve heuristic}}{\text{Time to reach optimal solution}} \times 100\%.$$

Table 24 shows these measurements of the heuristic applied to randomly generated test problems.

The computational results shown in Table 24 reflect the heuristic performance applied to building train routes and schedules from scratch. It is often referred as a *zero-based* train routing and scheduling approach. However, other than to build train routes and schedules from scratch, the model can also be used to solve other practical

Table 24: Computational Results of Train Heuristics

# Blocks	# Random problems	Average solution quality (% of optimal solution)	Average solution time (% of time to find optimal solution)
5	10	148%	63%
10	10	148%	30%
15	6	129%	37%
20	6	119%	7%

train problems. For example, the model can be used to determine the impacts of some changes of the current train schedules. Suppose we want to eliminate some trains from the system. Our model will examine how to reroute the blocks using those particular trains to other trains in the system. With a lot of variable fixed, this is not a hard problem to solve. Another example, given block information and all train schedules, the model will help us to find the trip-plan for the block and suggest the optimal block-swap locations (if any). The model can also determine an optimal schedule for given train routes and assignments. In this case, we have every integer variable fixed and the model will tell us the optimal timings for each trains. This will be used at least as a guide to derive optimal train schedules. Last but not least, as we illustrated earlier in our 63-block problem, this model can also be used to eliminate low-volume (daily or weekly) trains.

6.3 Notes on lower bounding problem

Similar to our sea cargo routing and scheduling model, we try to derive a bounding problem for our train routing and scheduling model. We derive an upper bounding problem to the model that empirically provides a bounding value stronger than the solution of the LP relaxation. The LP relaxation of the bounding problem is obtained in a relatively short time and tells us that the optimal profit from the sea cargo problem cannot be higher than this bounding value.

While it is our interest to derive a similar kind of bounding problem to our train

model, the train problem we deal with is a cost-minimizing problem and not a profit-maximizing problem. Here, we are interested in deriving a lower bounding value. However, due to the bilinear relations found in some constraints, we cannot easily apply the same idea as the way we derive an upper bounding problem to the sea cargo model. We cannot use the same tricks to find a closed form of the optimal train operating time and obtain a bounding problem.

Recall that the idea of how we introduce a bounding problem to the sea cargo model is to get rid of time variables and use the relations among bounds of the time variables to derive a closed form of optimal operating time for a fleet of ships. Now, our bilinear relations prohibit us using the same method because of the interdependent relation among them. For example, look at constraint (76) where we introduce variable $TD_{jsk} = T_{sk}XD_{jsk}$. Recall that H is the planning horizon and an equivalent linear reformulation of each TD_{jsk} is given by

$$\begin{aligned} TD_{jsk} &\geq 0, \\ TD_{jsk} &\geq T_{sk} + H * XD_{jsk} - H, \\ TD_{jsk} &\leq H * XD_{jsk}, \\ TD_{jsk} &\leq T_{sk}. \end{aligned}$$

To apply the same idea as how we derive a bounding problem for the sea cargo model, we have to analyze bounding relationships for T_{sk}, and one of the bound is

$$TD_{jsk} \geq T_{sk} + H * XD_{jsk} - H,$$

or after rearranging, we have

$$T_{sk} \leq TD_{jsk} - H * XD_{jsk} + H. \tag{82}$$

In the bounding problem for the sea cargo model, all bounding values of T_{sk} involve only variables other than time variables. Hence we may write $T_{sk} \leq f(x)$, for some function f of some fixed value x that does not depend on variable T_{sk}.

Table 25: Bounds for train problem

# Blocks	# Random problems	LP relaxation (% of optimal solution)	Lower bound performance (% of optimal solution)
5	10	47%	19%
10	10	36%	12%
15	6	51%	22%
20	6	49%	21%

However, in (82) we see that the right hand side is no longer independent of T_{sk} since $TD_{jsk} = T_{sk}XD_{jsk}$.

One may suggest to take out the constraints with bilinear relationship from the system, apply a similar idea to the way we get a bounding problem for the sea cargo problem, and still get a bound for the train model. This is absolutely true. However, since our systems have quite a number of constraints having bilinear terms, we end up taking out too many constraints resulting in a system that does not provide a lower bound as strong as expected. We include some computational results in Table 25 to show that the bound obtained by ignoring constraints having bilinear terms is empirically weaker than the LP relaxation of the original problem.

The performance of the lower bound is shown in the last column of Table 25 and is measured as the percentage of the optimal solution. Clearly, the closer this percentage value to 100%, the better. We may see that the lower bound is not strong enough compared to LP relaxation of the original problem.

CHAPTER VII

CONCLUSION AND FUTURE WORK

7.1 Conclusion

This research introduces new formulations to solve routing and scheduling problems, with the main application in answering actual problems faced by a sea-cargo shipping company in Asia and a railroad company in the United States. Our formulation better lends itself to extensions and related applications. The model reflects better situation in practice and perform well when dealing with extensive time aspects.

For the work in sea-cargo routing and scheduling, we focus our methodology on the *tramp* shipping operation. Tramp shipping is a demand-driven type of shipping operation. Tramp ships typically do not have fixed schedules, but are routed based on the pickup and download locations of profitable service requests. The problem is that given a set of products distributed among a set of ports, with each product having a pickup time window, download time window, and destination port, find the schedule for a fleet of ships that maximizes profit for a specified time horizon. The problem is modelled as a Mixed Integer Non-Linear Programming (MINLP) problem.

Our formulation involves quite a number of bilinear constraints that are linearized by taking the advantage of the linearization scheme for bilinear terms. It is a linearization scheme based on a convex and concave envelope approach allowing us to reformulate our model into an *equivalent* Mixed Integer Linear Program (MILP). We also exploit the special structure embedded in our model and uncover the totally-unimodular property embedded in some part of the activity matrix of multi-ship system.

We introduce three heuristic methods for solving sea-cargo routing and scheduling problems. The first heuristic (MPH) employs the idea of a rolling horizon concept.

Here we divide the time horizon into smaller time periods. We solve the earlier period and carry over the decision to the next period. By doing this, we simulate online cargo arrivals. The second heuristic (OSH) for our multi-ship problem utilizes the optimal solutions to one-ship problems. Here we permanently assign cargos that appear in only one ship and determine which ship should carry cargos that appear in more than one ship by creating a priority list. Based on the list, the model is again used to break any ties among competing ships. The third heuristic (SPH) utilizes the idea of common approaches in solving routing and scheduling problem; namely, a set partitioning approach. Since our problem has the flexibility of rejecting unprofitable cargos, the set packing approach is used. Here, we create a list of cargos based on their profitability and create columns *a priori* based on cargo combinations. Our model is again used to check if a cargo combination is feasible. If it is feasible, the solution of the model determines the optimum profit of that particular combination. In SPH we have a way to control the maximum number of columns generated by limiting the number of possible cargo combinations as well as the maximum number of cargos within each combination. If we allow the method to find all possible cargo combinations that can be served by each ship, we can eventually find the optimal solution. Extensive computational results are presented to show how good the heuristics perform in solving randomly generated problems. We also describe how these random test problems are generated. Generally, all proposed heuristic methods find solution that are within at least 80% of the optimal profit in most cases. OSH and SPH outperform MPH in terms of both solution quality and computational time.

We introduce an upper-bounding problem which our computational results show that the LP relaxation of the upper bounding problem gives bounds that are stronger than that given by the LP relaxation of the original problem. Also, those bounds are computed within 3 to 5 times faster than the time spent to find the LP relaxation of the original problem.

With a little modification, the model for our sea-cargo routing and scheduling

problem is readily extendable to reflect other practical needs in sea cargo operations. We introduce the notions of soft time windows with penalty and inter-ship cargo-transfer as extensions to our problem to show the advantages of our formulation of the problem formulation. In the soft time-window extension, we allow pickups and downloads to happen outside the predefined time-windows by imposing some penalty terms. Hence we will measure how far away a pickup or download happens outside its original time-windows. In the cargo-transfer extension, we allow a cargo to be picked-up and downloaded by different ships. We present some computational results on these extensions. The results show that more savings can be generated by including soft time-windows and inter-ship cargo-transfer capabilities to the model. Soft time-windows and cargo-transfer also allow more realistic situations in sea-cargo operation.

Another part of our work focuses on train routing and scheduling problem. We discuss the modification of the model to solve a train routing and scheduling problem faced by a real railroad company. Train routing and scheduling problems involve a lot of time aspects since one of the goals is to have optimal train schedules and timetables. Previous research in train routing and scheduling problems typically calls for separating the problem into several phases. Train route design problem, block-to-train assignment problem, and train optimal timetables are usually solved separately using an iterative procedure with feedbacks from one phase to another. Our model solves the routing and scheduling (timetable) problems altogether. Given a set of blocks to be carried from their origins to their destinations, our goal is to decide how many trains needed, construct train routes and schedules, and determine block-to-train assignments. We want the solution that minimizes the number of block transfers (block swaps) between trains, the number of trains used to serve all blocks, and some other efficiency measures used in train operation. Some computational experiments from real railroad data are presented and a bounding problem similar to that of ship's is briefly discussed.

Although this dissertation examined a range of important applications of our modeling technique, this approach is rather new and our research has identified a number of areas deserving future research. We present this future work in the following section.

7.2 Future Work

7.2.1 Sea cargo pricing strategy

We studied a demand-driven environment for tramp sea cargo operation. While many companies operating in this line of sea cargo operations deal with contract customers, many non-contract requests are received and provide significant additional potential revenues. With the limited number of ships owned by a company, determining whether a non-contract cargo will be served is critical, as well as determining the appropriate price charged for the cargo. The company knows that accepting a new cargo would affect the routes and schedules for some ships. The company has to know exactly which ship should pickup the new cargo and how much shipping fee should be charged. The multi-ship model can be used to determine the marginal shipping fee charged for the company to accept the new request. If the company has already had a predetermined list of shipping rates for particular origin-destination pairs, the model can be used to justify the profitability of a new request. If the company applies a dynamic pricing strategy, in which the price for non-contract cargos is determined by the position of the ships when the new request is received, the multi-ship model can also be used to determine how much below the standard shipping fee the price can be set to gain competitive advantage against the company's competitors.

7.2.2 Hub-and-spoke system in sea cargo operations

For middle to large tramp sea cargo companies, a hub and spoke system may be considered in their operation. Here, similar to the cargo-transfer extension discussed in Section 5.2, a cargo is allowed to be picked-up and downloaded by different ships with cargo transfers permitted at some predetermined ports. Typically, transfer ports

are significantly bigger ports and have strategic geographic locations, and they are often referred as *hubs*. The other smaller ports are referred to as *spokes*. Given the list of hubs and spokes, the multi-ship model with cargo transfer capability can be adapted and used to determine optimal routes and schedules of ships operating in a hub-and-spoke environment.

7.2.3 Performance of train routing and scheduling model

While some computational results of the train routing and scheduling model have been shown, there is a need to compare zero-based train routes and schedules generated by the model to the actual ones currently used by train companies. We can then analyze how well the model performs and how much savings can be generated. We can also compare the model performance to other modeling techniques used to answer train routing and scheduling problem. Performance of the train heuristic method can also be compared to other known heuristic methods.

APPENDIX A

RELAXATION OF THE PRODUCT OF VARIABLES

The purpose of this appendix is to explain the linearization technique used many times throughout the thesis. We find the technique is very useful in the formulation of our type. We use a lot of bilinear constraints and by special property of variables, the linearization is exact. This appendix demonstrates straightforward extensions and results of Al-Khayyal [3]. We also apply the idea to linearize a trilinear constraint we used in a sea-cargo routing and scheduling problem with extension to cargo-transfer.

Consider the compact set X defined by $x, y \in \Re$

$$A := \{(x, y) | l_x \leq x \leq u_x, l_y \leq y \leq u_y\}$$

We may see that the compact set A is formed by four constraints (i) $x - l_x \geq 0$, (ii) $y - l_y \geq 0$, (iii) $u_x - x \geq 0$, and (iv) $u_y - y \geq 0$.
Consider multiplying (i) and (ii)

$$\begin{aligned} & (x - l_x)(y - l_y) \geq 0 \\ \Rightarrow \quad & xy - l_y x - l_x y + l_x l_y \geq 0 \\ \Rightarrow \quad & xy \geq l_y x + l_x y - l_x l_y. \end{aligned}$$

Similarly, by multiplying (i) and (iv), (ii) and (iii), (iii) and (iv) we have

$$\begin{aligned} xy &\leq u_y x + l_x y - l_x u_y, \\ xy &\geq u_y x + u_x y - u_x u_y, \\ xy &\leq l_y x + u_x y - u_x l_y, \end{aligned}$$

respectively. Now consider the set B defined as

$$\begin{aligned} B := \{(x,y)| \quad & xy \geq l_y x + l_x y - l_x l_y, \\ & xy \leq u_y x + l_x y - l_x u_y, \\ & xy \geq u_y x + u_x y - u_x u_y, \\ & xy \leq l_y x + u_x y - u_x l_y\}. \end{aligned}$$

Notice that $A \subseteq B$. We also observe that the inequalities defining the set B form a system of linear relaxation to the bilinear term xy.

Proposition A.0.1 *If at least one of the variables x and y is binary, then the system of inequalities defining the set B is an* exact *linear reformulation to the bilinear term xy.*

Proof. If both x and y are binary, clearly $xy \in \{0,1\}$ and hence the linear reformulation is *exact*. Now, without loss of generality, let us suppose x is binary and $l_y \leq y \leq u_y$. Then there are two possible cases: $x = 0$ or $x = 1$. If $x = 0$ then substituting it to the system of inequalities defining the set B yields

$$\begin{aligned} B := \{(0,y)| \quad & xy \geq 0, \\ & xy \leq 0, \\ & xy \geq u_x y - u_x u_y, \\ & xy \leq u_x y - u_x l_y\} \end{aligned}$$

and by the first two inequalities, $xy = 0$, which is what it should be since $x = 0$. Similarly, if $x = 1$, then

$$\begin{aligned} B := \{(1,y)| \quad & xy \geq l_y, \\ & xy \leq u_y, \\ & xy \geq y, \\ & xy \leq y\}, \end{aligned}$$

or we may rewrite the system as $l_y \leq xy = y \leq u_y$, which is again what it should be since $x = 1$. Hence the proof is complete. ∎

Proposition A.0.1 is used to linearize bilinear terms introduced in our model formulation. All of bilinear constraints that we introduce in this thesis have the property that one of the variables is binary. We may further generalize the proposition to give an *exact* linear reformulation to trilinear constraints where at least two of the variables are binary.

Corollary A.0.1 *Let* $x, y \in \{0,1\}$ *and* $l_z \leq z \leq u_z$. *Then a system of exact linear reformulation of trilinear term* xyz *is given by*

$$\begin{aligned} xyz &\geq 0, \\ xyz &\geq u_z x + yz - u_z, \\ xyz &\leq u_z x, \\ xyz &\leq yz, \\ yz &\geq l_z y, \\ xy &\geq u_z y + z - u_z, \\ xy &\leq u_z y, \\ xy &\leq l_z y + z - l_z. \end{aligned}$$

Proof. First we apply Proposition A.0.1 to the terms y and z and we derive the last four constraints. Next observe that $0 \leq yz \leq u_z$, so we may apply Proposition A.0.1 one more time to the terms x and yz and derive the first four constraints. ∎

APPENDIX B

DUAL PROBLEM OF THE MILP RELAXATION OF SEA-CARGO MODEL

B.1 Objective function

$$(OBJ)\ \text{Minimize} \begin{cases} \sum_s \sum_k A_{sk} \\ + \sum_{i>0} \sum_s B_{is} + \\ \sum_s \sum_k VMAX_s \times LD_{sk} \\ \sum_{j \in \mathcal{U}} \sum_s \sum_{k<K} MN_{jsk} \\ - \sum_s \sum_{k<K} T_{adm} P_{sk} \\ + \sum_{j \in \mathcal{U}} Q_j \\ + \sum_i \sum_s \sum_k VX_{isk} \\ + \sum_i \sum_{l \neq i} \sum_s \sum_{k<K} VZ_{ilsk} \\ + \sum_j \sum_s VY_{js} \\ + \sum_{j \in \mathcal{U}} \sum_s \sum_k VXP_{jsk} \\ + \sum_j \sum_s \sum_k VXD_{jsk} \\ + \sum_j \sum_s \sum_k VXC_{jsk} \end{cases}$$

B.2 Dual Constraints related to Primal Variable X_{isk}

These type of dual constraints are complicated since they depend on whether a port i is a dummy port, a pick-up port (i.e. $i \in PP$), a download port (i.e. $i \in DP$), both pick-up port and download port, or neither (regular port).

Port i is a dummy port ($i = 0$)

$$A_{sk} + C_{sk} - DD_{isk} - T_{adm} P_{sk} + VX_{isk} \geq -PC_{is}, \qquad i = 0, s \in \mathcal{S}, k = 0$$

$$A_{sk}+C_{sk}-C_{s(k-1)}-DD_{isk}-E_{is(k-1)}-T_{adm}P_{sk}+VX_{isk} \geq -PC_{is}, \qquad i=0, s \in \mathcal{S}, k \in \mathcal{K}_s \backslash \{0, K_s\}$$

$$A_{sk}-C_{s(k-1)}-E_{is(k-1)}+VX_{isk} \geq -PC_{is}, \qquad i=0, s \in \mathcal{S}, k=K_s$$

Port i is a pick-up port ($i \in PP$)

$$A_{sk}+B_{is}-DD_{isk}-\sum_{j:i=PP_j, j \in \mathcal{U}} G_{jsk}+VX_{isk} \geq -PC_{is}, \qquad i \in PP, s \in \mathcal{S}, k=0$$

$$A_{sk}+B_{is}-DD_{isk}-E_{is(k-1)}-\sum_{j:i=PP_j, j \in \mathcal{U}} G_{jsk}+VX_{isk} \geq -PC_{is}, \qquad i \in PP, s \in \mathcal{S}, k \in \mathcal{K}_s \backslash \{0, K_s\}$$

$$A_{sk}+B_{is}-E_{is(k-1)}-\sum_{j:i=PP_j, j \in \mathcal{U}} G_{jsk}+VX_{isk} \geq -PC_{is}, \qquad i \in PP, s \in \mathcal{S}, k=K_s$$

Port i is a download port ($i \in DP$)

$$A_{sk}+B_{is}-DD_{isk}-\sum_{j:i=DP_j, j \in \mathcal{L} \cup \mathcal{U}} I_{jsk}+VX_{isk} \geq -PC_{is}, \qquad i \in PP, s \in \mathcal{S}, k=0$$

$$A_{sk}+B_{is}-DD_{isk}-E_{is(k-1)}-\sum_{j:i=DP_j, j \in \mathcal{L} \cup \mathcal{U}} I_{jsk}+VX_{isk} \geq -PC_{is}, \qquad i \in PP, s \in \mathcal{S}, k \in \mathcal{K}_s \backslash \{0, K_s\}$$

$$A_{sk}+B_{is}-E_{is(k-1)}-\sum_{j:i=DP_j, j \in \mathcal{L} \cup \mathcal{U}} I_{jsk}+VX_{isk} \geq -PC_{is}, \qquad i \in PP, s \in \mathcal{S}, k=K_s$$

Port i is both a pick-up and download port ($i \in PP$ and $i \in DP$)

$$A_{sk}+B_{is}-DD_{isk}-\sum_{j:i=PP_j, j \in \mathcal{U}} G_{jsk}-\sum_{j:i=DP_j, j \in \mathcal{L} \cup \mathcal{U}} I_{jsk}+VX_{isk} \geq -PC_{is}, \qquad i \in PP, s \in \mathcal{S}, k=0$$

$$A_{sk}+B_{is}-DD_{isk}-E_{is(k-1)}-\sum_{j:i=PP_j, j \in \mathcal{U}} G_{jsk}-\sum_{j:i=DP_j, j \in \mathcal{L} \cup \mathcal{U}} I_{jsk}+VX_{isk} \geq -PC_{is}, \qquad i \in PP, s \in \mathcal{S}, k \in \mathcal{K}_s \backslash \{0, K_s\}$$

$$A_{sk}+B_{is}-E_{is(k-1)}-\sum_{j:i=PP_j, j \in \mathcal{U}} G_{jsk}-\sum_{j:i=DP_j, j \in \mathcal{L} \cup \mathcal{U}} I_{jsk}+VX_{isk} \geq -PC_{is}, \qquad i \in PP, s \in \mathcal{S}, k=K_s$$

i is neither a pick-up or download port

$$A_{sk}+B_{is}-DD_{isk}+VX_{isk} \geq -PC_{is}, \qquad i \in PP, s \in \mathcal{S}, k=0$$

$$A_{sk}+B_{is}-DD_{isk}-E_{is(k-1)}+VX_{isk} \geq -PC_{is}, \qquad i \in PP, s \in \mathcal{S}, k \in \mathcal{K}_s \backslash \{0, K_s\}$$

$$A_{sk}+B_{is}-E_{is(k-1)}+VX_{isk} \geq -PC_{is}, \qquad i \in PP, s \in \mathcal{S}, k=K_s$$

B.3 Dual Constraints related to Primal Variable Z_{ilsk}

$$DD_{isk} + E_{lsk} - \frac{D_{il}}{24 * velocity} MD_{sk} + VZ_{ilsk} \geq 0, \qquad i,l \in \mathcal{I}, s \in \mathcal{S}, k \in \mathcal{K}_s \backslash \{K_s\}$$

B.4 Dual Constraints related to Primal Variable Y_{js}

Dual constraints related to primal variable Y_{js} can be separated into two types, those for $j \in \mathcal{L}$ and those for $j \in \mathcal{U}$.

$$-H_{js} + VY_{js} \geq SR_j, \qquad j \in \mathcal{L}, s \in \mathcal{S}$$

$$-F_{js} - H_{js} + J_{js} + VY_{js} \geq SR_j, \qquad j \in \mathcal{U}, s \in \mathcal{S}$$

B.5 Dual Constraints related to Primal Variable XP_{jsk}

Dual constraints related to primal variable XP_{jsk} are only defined for $j \in \mathcal{U}$.

$$F_{js} + G_{jsk} + kJ_{js} + KD_{jsk} + (M - LPT_j + 0.5T_{adm})N_{jsk} + (EPT_j + 0.5T_{adm} + \frac{V_j}{LR_j})O_{jsk} + ...$$

$$...+\frac{V_j}{LR_j} P_{sk} + VXP_{jsk} \geq 0, \qquad j \in \mathcal{U}, s \in \mathcal{S}, k \in \mathcal{K}_s \backslash \{0, K_s\}$$

$$F_{js} + G_{jsk} + kJ_{js} + VXP_{jsk} \geq 0, \qquad j \in \mathcal{U}, s \in \mathcal{S}, k = K_s$$

B.6 Dual Constraints related to Primal Variable XD_{jsk}

Dual constraints related to primal variable XP_{jsk} are different for $j \in \mathcal{L}$ and $j \in \mathcal{U}$.

$$H_{js} + I_{jsk} - KD_{jsk} + \frac{V_j}{DR_j} P_{sk} + VXD_{jsk} \geq 0, \qquad j \in \mathcal{L}, s \in \mathcal{S}, k \in \mathcal{K}_s \backslash \{K_s\}$$

$$H_{js} + I_{jsk} + VXD_{jsk} \geq -\frac{V_j}{DR_j} TCC_s, \qquad j \in \mathcal{L}, s \in \mathcal{S}, k = K_s$$

$$H_{js} + I_{jsk} + kJ_{js} - KD_{jsk} + \frac{V_j}{DR_j} P_{sk} + VXD_{jsk} \geq 0, \qquad j \in \mathcal{U}, s \in \mathcal{S}, k \in \mathcal{K}_s \backslash \{K_s\}$$

$$H_{js} + I_{jsk} + kJ_{js} + VXD_{jsk} \geq -\frac{V_j}{DR_j} TCC_s, \qquad j \in \mathcal{U}, s \in \mathcal{S}, k = K_s$$

B.7 Dual Constraints related to Primal Variable XC_{jsk}

$$KD_{jsk} + V_j LD_{sk} + VXC_{jsk} \geq 0, \qquad j \in \mathcal{L} \cup \mathcal{U}, s \in \mathcal{S}, k = 0$$

$$KD_{jsk} - KD_{js(k-1)} + V_j LD_{sk} + VXC_{jsk} \geq 0, \qquad j \in \mathcal{L} \cup \mathcal{U}, s \in \mathcal{S}, k \in \mathcal{K}_s \backslash \{0, K_s\}$$

$$-KD_{js(k-1)} + V_j LD_{sk} + VXC_{jsk} \geq 0, \qquad j \in \mathcal{L} \cup \mathcal{U}, s \in \mathcal{S}, k = K_s$$

B.8 Dual Constraints related to Primal Variable T_{sk}

$$\sum_{j \in \mathcal{U}} N_{jsk} + P_{sk} \geq -FC_s(24)(v_s), \qquad s \in \mathcal{S}, k = 0$$

$$-\sum_{j \in \mathcal{U}} -P_{s(k-1)} \geq -TCC_s, \qquad s \in \mathcal{S}, k \in \mathcal{K}_s \backslash \{0, K_s\}$$

$$\sum_{j \in \mathcal{U}} N_{jsk} - \sum_{j \in \mathcal{U}} O_{js(k-1)} + P_{sk} - P_{s(k-1)} \geq 0, \qquad s \in \mathcal{S}, k = K_s$$

B.9 Dual Constraints related to Primal Variable TT_{sk}

$$MD_{sk} + \sum_{j \in \mathcal{U}} O_{jsk} + P_{sk} \geq -FC_s(24)(velocity), \qquad s \in \mathcal{S}, k \in \mathcal{K}_s \backslash \{K_s\}$$

REFERENCES

[1] AHUJA, R.K., CUNHA, C.B., SAHIN, G., "Network Models in Railroad Planning and Scheduling," 2005.

[2] AL-KHAYYAL, F. and HWANG, S.J., "Inventory constrained maritime routing and scheduling for multi-commodity liquid bulk, Part I: Applications and model," *European Journal of Operation Research*, Article in Press, 2005.

[3] AL-KHAYYAL, F., "Generalized Bilinear Programming: Part 1. Models, Applications, and Linear Programming Relaxation," *European Journal of Operation Research*, vol. 60, pp. 309-314, 1992.

[4] AL-KHAYYAL, F. and FALK, J., "Jointly Constrained Biconvex Programming," *Mathematics of Operation Research*, vol. 8, pp. 273-286, 1983.

[5] ANBIL, R., TANGA, R., JOHNSON, E.L., "A Global Approach to Crew-Pairing Optimization", *IBM Systems Journal*, vol. 31, no. 1, pp. 71-78, 1992.

[6] APPELGREN, L.H., "A Column Generation Algorithm for a Ship Scheduling Problem," *Transportation Science*, vol. 3, pp. 53-68, 1969.

[7] ASSAD, A., "Models for rail transportation," *Transportation Research, Part A: Policy and Practice*, vol. 14, pp. 205-220, 1980.

[8] BALAKRISHNAN, N., "Simple Heuristics for the Vehicle Routing Problme with Soft Time Windows," *Journal of Operation Research Society*, vol. 44, no. 3, pp. 279-287, 1993.

[9] BAUSCH, D.O., BROWN, G.G., RONEN, D., "Scheduling short-term marine transport of bulk products," *Maritime Policy and Management*, vol. 25, no.4, pp. 335348, 1998.

[10] BAZARAA, M.S., SHERALI, H. and SHETTY, C., *Nonlinear Programming: Theory and algorithms, 2nd*, New York, Wiley, 1993.

[11] BERTSIMAS, D., TSITSIKLIS, J.N., *Introduction to Linear Optimizatoin.* Athena Scientific, Belmont, Massachusetts, 1997.

[12] CHRISTIANSEN, M., FAGERHOLT, K., RONEN, D., "Ship Routing and Scheduling: Status and Perspective," *Transportation Science*, vol. 38, No. 1, pp. 1- 18, 2004.

[13] CHRISTIANSEN, M., FAGERHOLT, K., NYGREEN, B., RONEN, D., "Maritime Transportation," Technical Reports, Norwegian University of Science and Technology, Trondheim, Norway, 2003.

[14] CHRISTIANSEN, M., FAGERHOLT, K., NYGREEN, B., RONEN, D., "Ship Routing and Scheduling: Status and Perspectives," *Transportation Science*, vol. 38, no. 1, pp. 118, 2004.

[15] CHRISTIANSEN, M., FAGERHOLT, K., "Robust Ship Scheduling with Multiple Time Windows," *Naval Research Logistics*, vol. 49, pp. 611-625, 2002.

[16] CHRISTIANSEN, M., "Decomposition of a combined inventory and time constrained ship routing problem," *Transportation Science*, vol. 33, no.1, pp. 316, 1999.

[17] CORDEAU, J.F., Toth, P., and VIGO, D., "A Survey of Optimization Models for Train Routing and Scheduling," *Transportation Science*, vol. 32, no. 4, pp. 380-404, 1998.

[18] CRAINIC, T.G., FERLAND, J.A., and ROUSSEAU, J.M., "A Tactical Planning Model for Rail Freight Transportation," *Operational Research '84*, J.P. Brans (ed.), Elsevier Science Publishers, Amsterdam, pp. 707-720, 1984.

[19] DANSKIN, J.M., *The Theory of Max-Min*. Springer-Verlag New York Inc., 1967.

[20] DUMAS, Y., DESROSIERS, J., SOUMIS, F., "The Pickup and Delivery Problem with Time Windows," *European Journal of Operation Research*, vol. 54, pp. 7-22, 1991.

[21] FAGERHOLT, K., "A Computer-based Decision Support System for Vessel Fleet Scheduling - Experience and Future Research," *Decision Support Systems*, vol. 37, pp. 35-47, 2004.

[22] FAGERHOLT, K., "Ship scheduling with soft time windows - an optimization based approach," *International Transactions in Operation Research*, vol. 6, no. 5, pp. 453-464, 1999.

[23] FISCHETTI, M., and TOTH, P., "An Additive Bounding Procedure for Combinatoral Optimization Problems," *Operations Research*, vol. 37, pp. 319-328, 1989.

[24] FISHER, M.L., ROSENWEIN, M.B., "An interactive optimization system for bulk-cargo ship scheduling," *Naval Research Logistics*, vol. 36, pp. 27-42, 1988.

[25] GORMAN, M.F., "An application of genetic and tabu searches to the freight railroad operating plan problem," *Annals of Operation Research*, vol. 78, pp. 51-69, 1998.

[26] HAGHANI, A.E., "Formulation and solution of a combined train routing and makeup, and empty car distribution model," *Transportation Research, Part B: Methodological*, vol. 23, no. 6, pp. 433-452, 1989.

[27] HEIDELOFF, C., STOCKMANN, D., *ISL Market Analysis 2005*, SSMR, 2005.

[28] HU, J., JOHNSON, E.L., "Computational Results with a Primal-Dual Subproblem Simplex Method", *Operations Research Letters*, vol. 25, pp. 149-157, 1999.

[29] HUNTLEY, C.L., BROWN, D.E., SAPPINGTON, D.E., and MARKOWICZ, B.P., "Freight Routing and Scheduling at CSX Transportation," *Interfaces*, vol. 25, no. 3, pp. 58-71, 1995.

[30] HWANG, S.J., "Inventory Constrained Maritime Routing and Scheduling for Multi-Commodity Liquid Bulk," PhD Thesis, Georgia Institute of Technology, 2005.

[31] JAW, J.J., ODONI, A.R., PSARAFTIS, H.N., and WILSON, N.H.M., "A Heuristic Algoritm for the Multi-vehicle Advance Request Dial-a-ride Problem with Time Windows," *Transportation Research Part B*, vol. 20B, pp. 243-257, 1986.

[32] JETLUND, A. and KARIMI, I., "Improving the logistics of multi-compartment chemical tankers," *Computers and Industrial Engineer*, vol. 33, pp. 689-692, 1997.

[33] JOHNSON, E. and BARNES, E., "Computational Optimization: LaGrange Methods and Decomposition," Lecture Notes, 2002.

[34] KALANTARI, B., HILL, A.V., and ARORA, S.R., "An Algorithm for the Traveling Salesman Problem with Pickup and Delivery Customers," *European Journal of Operation Research*," vol. 22, pp. 377-386, 1985.

[35] KEATON, M.H., "Designing optimal railroad operating plans - Lagrangian-relaxation and heuristic approaches," *Transporation Research, Part B: Methodological*, vol. 23, no. 6, pp. 415-431, 1989.

[36] KEATON, M.H., "Designing railroad operating plans - A dual adjustment method for implementing lagrangian-relaxation," *Transportation Science*, vol. 26, no. 4, pp. 263-279, 1992.

[37] KIM, S.H., LEE, K.K., "An optimization-based decision support system for ship scheduling," *Computers and Industrial Engineering*, vol. 33, no. 3-4, pp. 689-692, 1997.

[38] KOLEN, A.W.J., RINNOOY KAN, A.H.G., TRIENEKENS, H.W.J.M., "Vehicle Routing with Time Windows," *Operations Research*, vol. 35, no. 2, pp. 266-273, 1987.

[39] KOSKOSIDIS, Y.A., POWELL, W.B., SOLOMON, M.M., "An Optimization-Based Heuristic for Vehicle Routing and Scheduling with Soft Time Window Constraints," *Transportation Science*, vol 26, no. 2, pp. 69-85, 1992.

[40] LAMPORT, L., *LATEX User's Guide and Reference Manual*. Addison-Wesley Publishing Company, Inc., 1986.

[41] LAU, H.C., LIANG, Z., "Pickup and Delivery with Time Windows: Algorithms and Test Case Generation," *International Journal on artificial Intelligence Tools*, vol. 11, no. 3, pp. 455-472, 2002.

[42] LAU, H.C., SIM, M., TEO, K.M., "Vehicle Routing Problem with Time Windows and A Limited Number of Vehicles," *European Journal of Operation Research*, vol. 148, pp. 559-569, 2003.

[43] LITTLE, J., MURTY, K., SWEENEY, D., and KAREL, C., "An Algorithm for the Traveling Salesman Problem," *Operation Research*, vol. 11, pp. 972-989, 1963.

[44] LU, Q., and DESSOUKY, M., "An Exact Algorithm for the Multiple Vehicle Pickup and Delivery Problem," *Transportation Science*, vol. 38, pp. 503-514, 2004.

[45] MORLOK, E.K., and PETERSON, R.B., "Final Report on a Development of a Geographic Transportation Network Generation and Evaluation Model," *Journal of Transporation Research Forum*, vol. 11, pp. 71-105, 1970.

[46] NANRY, W.P., and BARNES, J.W., "Solving the Pickup and Delivery Problem with Time Windows Using Reactive Tabu Search," *Transporation Research Part B*, vol. 34, pp.107-121, 2000.

[47] NEMHAUSER, G. and WOLSEY, L., *Integer and Combinatorial Optimization*. New York, NY: John Wiley & Sons, 1998.

[48] NOWAK, M.A., "The Pickup and Delivery Problem with Split Loads," PhD Thesis, Georgia Institute of Technology, 2005.

[49] PSARAFTIS, H., "Foreword to the focused issue on maritime transportation," *Transportation Science*, vol. 33, 1999.

[50] PSARAFTIS, H., "Dynamic vehicle routing problems," in B. L. Goldern, A.A. Assad, *Vehicle Routing: Methods and studies*, North-Holland: Elsevier, pp. 223-248, 1988.

[51] PSARAFTIS, H., "A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem," *Transportation Science*, vol. 14, no. 2, pp. 130-154, 1980.

[52] RANA, K. and VICKSON, R.G., "Routing Container Ships Using Lagrangean Relaxation and Decomposition," *Transportation Science*, vol. 25, No. 3, pp. 201-214, 1991.

[53] RONEN, D., "Ship Scheduling: The Last Decade," *European Journal of Operation Research*, vol. 71, pp. 325-333, 1993.

[54] RONEN, D., "Short-term scheduling of vessels for shipping bulk or semi-bulk commodities originating in a single area," *Operation Research*, vol. 34, No. 1, pp. 164-173, 1986.

[55] RONEN, D., "Cargo Ship Routing and Scheduling: Survey of Models and Problems," *European Journal of Operation Research*, vol. 12, pp. 119-126, 1983.

[56] SAVELSBERGH, M.W.P., and SOL, M., "General Pickup and Delivery Problem," *Transporation Science*, vol. 29, no. 1, pp. 17-29, 1995.

[57] SAVELSBERGH, M.W.P., and SOL, M., "Drive: Dynamic Routing of Independent Vehicles," *Operations Research*, vol. 46, pp. 474-490, 1998.

[58] SCHRIJVER, A., *Theory of Linear and Integer Programming*. Baffins Lane-Chichester, John Wiley & Sons, 1986.

[59] SCOTT, J.L., "A transportation model, its development and application to a ship scheduling problem," *Asia-Pacific Journal of Operation Research*, vol. 12, pp. 111-128, 1995.

[60] SEXTON, T.R., and BODIN, L.D., "Optimizing Single Vehicle Many-to-many Operations with Desired Delivery Times: I. Scheduling," *Transportation Sciences*, vol. 19, pp. 378-410, 1985.

[61] SEXTON, T.R., and BODIN, L.D., "Optimizing Single Vehicle Many-to-many Operations with Desired Delivery Times: I. Routing," *Transportation Sciences*, vol. 19, pp. 411-435, 1985.

[62] SHAPIRO, J.F., *Mathematical Programming: Structures and Algorithms*. John Wiley & Sons, Canada, 1979.

[63] SHERALI, H.D., AL-YAKOOB, S.M., HASSAN, M.M., "Fleet management models and algorithms for an oil tanker routing and scheduling problem," *IIE Transportation*, vol. 31, pp.395406, 1999.

[64] SOLOMON, M.M., "Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints," *Operations Research*, vol. 35, no. 2, pp. 254-265, 1987.

[65] SOLOMON, M.M., DESROSIERS, J., "Survey Paper: Time Window Constrained Routing and Scheduling Problems," *Transportation Science*, vol. 22, no. 1, pp. 1-13, 1988.

[66] THOMET, M.A., "A User-Oriented Freight Railroad Operating Policy," *IEEE Trans. Systems, Man Cybernet*, vol. 1, pp. 349-356, 1971.

[67] TOTH, P., and VIGO, D., "Heuristic Algorithms for the Handicapped Persons Transportation Problem," *Transportation Science*, vol. 31, no. 1., pp. 60-71, 1997.

[68] VAN DER BRUGGEN, L.J.J., LENSTRA, J.K., and SCHUUR, P.C., "Variable-depth Search for the Single-vehicle Pickup and Delivery Problem with Time Windows," *Transportation Science*, vol. 27, pp. 298-311, 1993.

VITA

Aang Daniel was born on February 7, 1977 in Tasikmalaya, Jawa Barat, Indonesia. He received his Bachelor of Science in Industrial Engineering from Bandung Institute of Technology, Bandung, Indonesia in Fall 1998. Prior to continuing his education, he worked for Accenture for 2.5 years where he served as a Business and Management Consultant. He joined the graduate program in the School of Industrial and Systems Engineering at Georgia Tech in Fall 2001. Mr. Daniel's research interests are primarily in applying optimization methods to answer problems found in transportation industries.